VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Virtuální sdílená tabule

Whiteboard Online Sharing Tool

David Lajtoch

Bakalářská práce

Vedoucí práce: Ing. Svatopluk Štolfa, Ph.D.

Ostrava, 2021

Abstrakt

Práce popisuje tvorbu webové aplikace virtuální online sdílené tabule, porovnání ostatních aplikací tohoto typu, použité technologie, architekturu a nejdůležitější skripty v ní obsažené. Nástroj online sdílení virtuálních tabulí slouží ke společnému kreslení uživatelů na virtuální plátno za pomoci prohlížeče, kde mohou nanášet čáry, vkládat text, obrázky a další objekty. Aplikace může sloužit k zábavě, nebo jako pomůcka pro vizuální vysvětlení nějakého problému.

Klíčová slova

virtuální online sdílená tabule, webová aplikace, volné kreslení, ASP.NET Core, SignalR, JavaScript

Abstract

The thesis describes the creation process of a web application focused on the online sharing of virtual whiteboards. Further, it compares other existing and similar applications, analyzes used technologies, architecture and the most important scripts used. Whiteboard online sharing tools allow users to draw on virtual canvas using a web browser. Users can draw lines, write text, add images and other objects. This application can be used for fun or to visually describe various problems.

Keywords

whiteboard online sharing tool, web application, free drawing, ASP.NET Core, SignalR, JavaScript

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Svatopluku Štolfovi, Ph.D., za rady, které mi poskytl při její tvorbě.

Obsah

Seznam použitých symbolů a zkratek **6**

Seznam obrázků **7**

Seznam tabulek **8**

1 Úvod **9**

2 Porovnání nástrojů sdílených tabulí **10**

2.1 Obsáhlejší aplikace 12

2.2 Jednodušší aplikace 13

2.3 Výsledek porovnání 14

3 Funkční specifikace vlastní aplikace **15**

3.1 Seznam funkčností 15

3.2 Use Case Diagram 16

3.3 Popis případů užití 17

4 Použité technologie **20**

4.1 HTML - HyperText Markup Language 20

4.2 CSS - Cascading Style Sheets 20

4.3 Less - Leaner Style Sheets 20

4.4 JavaScript 21

4.5 jQuery 21

4.6 Fabric.js 21

4.7 C# 21

4.8 ASP.NET Core 22

4.9 ASP.NET Core SignalR 22

4.10 Ostatní využité knihovny a moduly 23

4.11 Další možnosti 24

5 Architektura a struktura aplikace **25**
5.1 ASP.NET Core MVC Architektura . . . 25
5.2 Role komponentů ASP.NET Core MVC . . . 26
5.3 Třídní diagram . . . 26
5.4 Struktura aplikace . . . 27
5.5 Struktura databáze . . . 28
5.6 Lineární zápis databáze . . . 28
5.7 Datový model databáze . . . 29

6 Implementace funkcí a uživatelské rozhraní **30**
6.1 Uživatelské rozhraní . . . 30
6.2 Obecný popis jednotlivých skriptů . . . 33
6.3 Postup volání skriptů . . . 35
6.4 Bližší popis skriptů . . . 39

7 Zhodnocení a závěr **54**

Literatura **55**

Přílohy **56**

A Třídní diagram aplikace **57**

B Návod ke spuštění aplikace **59**

C Manuál k aplikaci **60**
C.1 Okno volby přihlášení . . . 61
C.2 Okno přihlášení registrovaného uživatele . . . 62
C.3 Okno registrace . . . 63
C.4 Nabídka možností . . . 64
C.5 Nabídka nástrojů . . . 65
C.6 Nabídka barev . . . 66
C.7 Nabídka velikosti čáry / textu . . . 67
C.8 Nabídka obrázku . . . 68
C.9 Okno uložení / načtení tabule . . . 69
C.10 Způsob používání aplikace . . . 70

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous JavaScript and XML
API	–	Application Programming Interface
ASP	–	Active Server Pages
CSHTML	–	C Sharp HyperText Markup Language
CSS	–	Cascading Style Sheets
DOM	–	Document Object Model
ER	–	Entity Relationship
HTML	–	HyperText Markup Language
JSON	–	JavaScript Object Notation
Less	–	Leaner Style Sheets
MVC	–	Model View Controller
PDF	–	Portable Document Format
UC	–	Use Case
URL	–	Uniform Resource Locator
XML	–	Extensible Markup Language

Seznam obrázků

3.1 Use Case Diagram aplikace 16

5.1 Společné fungování modelu, pohledu a kontroleru 25
5.2 ER diagram databáze aplikace 28

6.1 Přihlašovací okno zobrazené při prvním spuštění aplikace 30
6.2 Kreslící plátno a nabídka nástrojů 31
6.3 První postup volání skriptů 35
6.4 Druhý postup volání skriptů 37
6.5 Sekvenční diagram - přihlášení registrovaného uživatele 40
6.6 Sekvenční diagram - přidání objektu na tabuli 46
6.7 Sekvenční diagram - uložení tabule do databáze 50

A.1 Třídní diagram aplikace 58

C.1 Okno volby přihlášení 61
C.2 Okno přihlášení registrovaného uživatele 62
C.3 Okno přihlášení registrovaného uživatele 63
C.4 Nabídka možností 64
C.5 Nabídka nástrojů 65
C.6 Nabídka barev 66
C.7 Nabídka velikosti čáry / textu 67
C.8 Nabídka obrázku 68
C.9 Okno uložení / načtení tabule 69

Seznam tabulek

2.1 Porovnání funkčnosti existujících aplikací online sdílených tabulí 11

5.1 Datový model tabulky **User** 29
5.2 Datový model tabulky **Whiteboard** 29

Kapitola 1

Úvod

Při práci na skupinových projektech, komunikaci s kolegy, nebo při nutnosti ostatním vysvětlit určité téma, se často vyskytne potřeba využít nějakého vizuálního zobrazení, náčrtů, které mohou v dané situaci usnadnit práci, zvláště pokud probíhá distančně. Tyto situace často nastávají spontánně, kdy je třeba využít jednoduchého a rychlého nástroje, který nevyžaduje žádnou jeho předchozí znalost. Řešením těchto situací tedy mohou být programy / aplikace, které se zaměřují na online sdílení virtuálních tabulí. Takovéto virtuální tabule jsou v základu obdobou fyzických tabulí, lze na ně volně kreslit, avšak nabízejí i mnohem složitější funkce, které by na skutečné tabuli byly velice zdlouhavé, nebo vyloženě nemožné.

Je jasné, že by bylo ideální, kdyby tyto aplikace nebylo nutné instalovat a mohly by tedy běžet přímo v prohlížeči. Dnešní technologie tvorbu takovýchto aplikací umožňují, uživatel si může zvolit z velké nabídky kreslících aplikací. Většina z nich má mnoho funkcí, vyžadují registraci, jsou vyvíjeny dlouhou dobu, velkými týmy. Přestože je velký počet funkcí obecná výhoda aplikace, může také vést k odrazení uživatele, který hledá něco rychlého, jednoduchého a použitelného bez jakéhokoli zdržování. Aplikace s těmito rysy je výsledkem této bakalářské práce.

Kapitola 2

Porovnání nástrojů sdílených tabulí

Prvním krokem při tvorbě sdílené virtuální tabule bylo zjistit, jaké jiné aplikace zabývající se online sdílenou tabulí již existují a jaké mají funkce a vlastnosti. Do výběru porovnávaných aplikací jsou zařazeny pouze ty aplikace, které jsou schopny běžet přímo v prohlížeči, bez nutnosti instalace.

Z desítek dostupných aplikací bylo vybráno celkem šest aplikací, které se řadí mezi nejpopulárnější a jejich porovnání by mělo stačit k vytvoření představy o tom, co by aplikace tohoto typu měla obsahovat [1]. Těchto šest aplikací bylo také rozděleno na dvě skupiny - obsáhlejší aplikace, které se nesnaží příliš zaměřovat na nenáročné uživatele, ale spíše na uživatele, kteří potřebují řešit komplikovanější problémy, a na jednodušší aplikace směřující k nenáročným uživatelům, nabízející menší množství funkcí, avšak kladoucí méně nároků na uživatele a jsou jednodušší na ovládání.

1. **Obsáhlejší aplikace**
   (a) Miro
   (b) Stormboard
   (c) Limnu
2. **Jednodušší aplikace**
   (a) Whiteboard Fox
   (b) Microsoft Whiteboard
   (c) Whiteboard

Každá aplikace byla otestována a vyzkoušena, přičemž byly zaznamenány její funkce, přednosti a zaměření. Užitečné funkce a funkce, které se u daných aplikací vyskytovaly častěji, byly zaznamenány do tabulky 2.1 (str. 11), pomocí které lze aplikace jasně porovnat.

Tabulka 2.1: Porovnání funkčnosti existujících aplikací online sdílených tabulí

Funkce	**Miro**	**Stormboard**	**Limnu**	**Whiteboard Fox**	**MS Whiteboard**	**Whiteboard**
Přístupnost						
Nutná registrace	ANO	ANO	ANO	NE	ANO	NE
Kompletně zdarma	NE	NE	NE	NE	ANO	ANO
Formát tabule						
Nekonečná tabule	ANO	ANO	ANO	ANO	ANO	ANO
Šablony	ANO	ANO	NE	NE	NE	NE
Nástroje						
Práce s objekty	ANO	ANO	ANO	ANO	ANO	NE
Transformace objektů	ANO	ANO	ANO	NE	ANO	NE
Spojování objektů šipkami	ANO	NE	NE	NE	NE	NE
Volné kreslení	ANO	ANO	ANO	ANO	ANO	ANO
Vložení textu	ANO	ANO	ANO	ANO	ANO	ANO
Vytvoření pozn. listu	ANO	NE	ANO	NE	ANO	NE
Vložení komentáře	ANO	NE	NE	NE	NE	NE
Přidání geom. tvaru	ANO	ANO	ANO	NE	NE	NE
Vložení obrázku	ANO	ANO	ANO	ANO	NE	NE
Vložení text. souboru	ANO	NE	ANO	NE	NE	NE
Vložení iframe objektu	ANO	NE	NE	NE	NE	NE
Vložení tabulky	ANO	NE	NE	NE	NE	NE
Vložení grafu	ANO	NE	NE	NE	NE	NE
Ostatní funkce						
Komentáře mimo tabuli	ANO	ANO	NE	NE	NE	NE
Chat	ANO	ANO	ANO	NE	NE	NE
Video chat	ANO	NE	ANO	NE	NE	NE
Ankety	ANO	NE	NE	NE	NE	NE
Viditelné kurzory uživatelů	ANO	ANO	ANO	NE	NE	NE
Historie akcí	ANO	ANO	ANO	ANO	NE	ANO
Rozpoznání tvarů	NE	NE	ANO	NE	NE	ANO
Uložení tabule	ANO	ANO	ANO	ANO	ANO	NE
Export jako obrázek	ANO	ANO	ANO	ANO	ANO	ANO

2.1 Obsáhlejší aplikace

2.1.1 Miro

Z testování jasně vychází jako aplikace s největším počtem funkcí. Miro se zaměřuje na náročnější uživatele a vývojové týmy, jelikož vedle obyčejnějších funkcí (volné kreslení, vkládání obrázků a mnoho dalších) nabízí také možnost využití šablon, vytváření snímků k prezentaci s prezentačním režimem a sdílení obrazovky. Jako jediná z vybraných aplikací také nabízí funkci exportování tabule do formátu PDF. Aplikace umožňuje také využití přídavných modulů dále rozšiřujících její funkce [2].

2.1.2 Stormboard

Oproti ostatním testovaným aplikacím se liší tím, že tabule je rozdělena na dvě části - první (nazývaná Storm), slouží jako rozvrh sekcí, přičemž do každé sekce lze přidávat elementy a rozdělovat k nim úkoly a nápady. Každý element pak tvoří druhou část tabule. Jedním z těchto elementů je samotné kreslící plátno, na které bylo zaměřeno testování funkcí. Z tohoto rozvržení aplikace na dvě části vyplývá, že Stormboard necílí na běžné uživatele, ale na pracovní týmy [3].

2.1.3 Limnu

Mimo běžnějších funkcí umožňuje aplikace Limnu rozdělit různým uživatelům administrativní práva k daným prvkům tabule, nebo schopnost rozpoznávání tvarů nakreslených uživatelem v režimu volného kreslení. Pokročilejší funkce jsou také oproti ostatním aplikacím více skryté, nezatěžují tedy běžného uživatele, na kterého vedle pokročilejších uživatelů míří [4].

2.2 Jednodušší aplikace

2.2.1 Whiteboard Fox

Whiteboard Fox se řadí mezi jednodušší aplikace ze seznamu, avšak kromě základních funkcí jako je volné kreslení, kreslení přímek a vkládání obrázků, nabízí také zajímavější funkce - rozdělení práv účastníků na úpravu a mazání objektů, zkopírování obsahu tabule do nové a nebo tmavý režim, který je však přístupný pouze uživatelům se zakoupeným předplatným. Whiteboard Fox cílí na uživatele, kteří nevyžadují funkce sloužící vývojovým týmům, ale využití nalezne například pro online výuku [5].

2.2.2 Microsoft Whiteboard

Aplikace Microsoft Whiteboard je dostupná jako internetová aplikace, ale také jako desktopová aplikace. Tyto dvě verze se od sebe výrazně liší v počtu funkcí, kde internetová obsahuje pouze velice základní funkce a desktopová nabízí funkce na úrovni aplikací ze skupiny obsáhlejších aplikací. Testování bylo zaměřeno na internetovou verzi aplikace. V porovnání s ostatními aplikacemi kromě opravdu velice jednoduchého uživatelského rozhraní nenabízí příliš výhod nebo zajímavých funkcí, chybí například i základní funkce vkládání obrázků [6].

2.2.3 Whiteboard

Aplikace s jednoduchým názvem Whiteboard se také zaměřuje na nenáročné uživatele. Jako jediná z porovnávaných aplikací nepracuje s objekty, nelze tedy nakreslené tvary nebo texty dále upravovat nebo přesouvat, umožňuje však využít nástroj gumy k přesnému smazání částí tabule namísto celých objektů. Poněkud překvapivě je však podporována pokročilejší funkce rozpoznávání nakreslených tvarů nebo tmavý režim. Zaujme hlavně svým sympatickým a jednoduchým uživatelským rozhraním [7].

2.3 Výsledek porovnání

Z výsledné tabulky 2.1 (str. 11) lze především vyčíst nejvíce používané funkce, které vybrané aplikace nabízí. Mezi často používané funkce byly zařazeny funkce, které nabízelo alespoň pět ze šesti vybraných aplikací:

1. **Nekonečná tabule** - tabule není omezena formátem a dokáže pojmout velké množství objektů.
2. **Práce s objekty** - tabule funguje na základě objektů, které uživatel na tabuli přidává. Výhoda tohoto přístupu spočívá v možnosti dále s objekty pracovat (měnit jejich velikost, přesouvat je, mazat celé objekty). Nevýhodou tohoto přístupu je nemožnost částečně "gumovat"objekty.
3. **Volné kreslení** - umožňuje uživateli volně kreslit čáry jedním nebo více nástroji.
4. **Vložení textu** - je možné psát a vkládat text, který lze později upravovat.
5. **Historie akcí** - uživatel má možnost navrátit akce, které provedl.
6. **Uložení tabule** - tabuli lze uložit do databáze aplikace pro pozdější načtení, u testovaných aplikací vyžaduje registraci uživatele.
7. **Export jako obrázek** - Aplikace je schopná vytvořit z nakreslených objektů obrázkový soubor pro stažení do počítače.

Tyto funkce se tedy dají považovat za funkce, které by měla obsahovat každá aplikace sdílených tabulí, aby byla schopná konkurovat ostatním. Aplikace, která je výsledkem této bakalářské práce, je nabízí všechny, chybí pouze funkce historie akcí, která při vývoji aplikace dala přednost funkci vkládání obrázku do tabule. Přidání funkce historie akcí by v budoucím vývoji neměla představovat komplikace, jelikož je aplikace připravena na co nejjednodušší budoucí rozšíření.

Kapitola 3

Funkční specifikace vlastní aplikace

3.1 Seznam funkčností

3.1.1 Správa uživatelů

1. Registrace uživatele
2. Přihlášení uživatele jako hosta
3. Přihlášení uživatele jako registrovaného

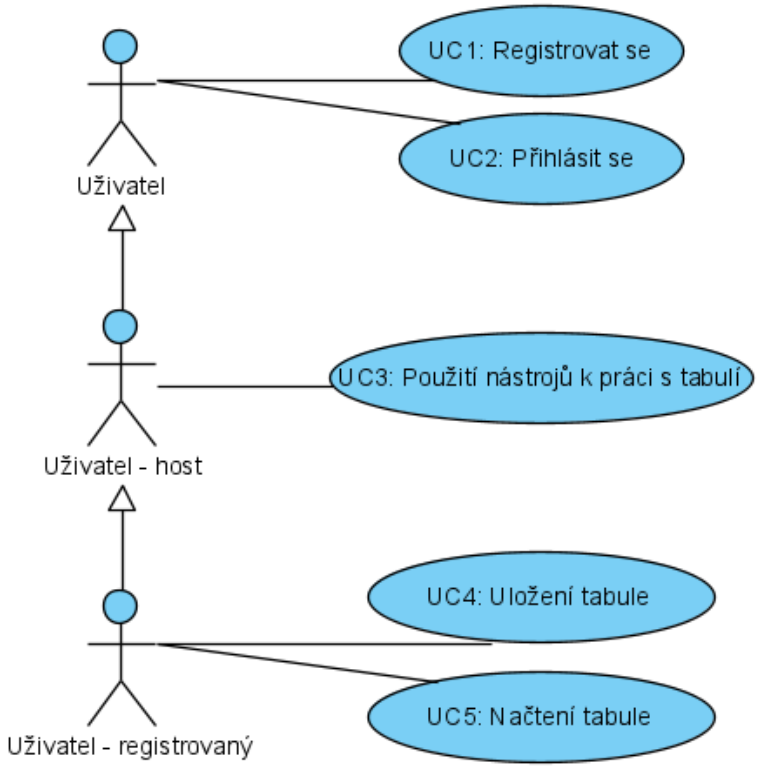
3.1.2 Použití nástrojů k práci s tabulí

1. Vytvoření nové sdílející skupiny a odkazu sdílení
2. Rychlé zkopírování odkazu sdílení do schránky
3. Nástroj mazání objektů
4. Nástroj výběru / transformace objektů
5. Nástroj vkládání textu
6. Nástroj volného kreslení
7. Nástroj změny barvy
8. Nástroj změny tloušťky čáry
9. Přiblížení / oddálení pohledu na tabuli
10. Vložení obrázku na tabuli
11. Uložení aktuálního pohledu jako obrázku

3.1.3 Správa tabulí uživatele

1. Uložení tabule
2. Načtení tabule

3.2 Use Case Diagram



Obrázek 3.1: Use Case Diagram aplikace

3.3 Popis případů užití

3.3.1 UC1: Registrovat se

Popis: uživatel si vytvoří profil, díky kterému získá přístup k dalším funkcím.
Aktér: uživatel.

Prekondice: uživatel má zobrazené registrační okno.
Postkondice: uživatel má vytvořený profil.

Scénář:

1. Uživatel vyplní potřebné údaje.
2. Uživatel potvrdí údaje tlačítkem.
3. Systém zkontroluje správnost zadaných údajů.
   (a) Uživatel zadal nesprávné údaje.
      i. Systém zobrazí upozornění.
      ii. Bod 1.
4. Uživateli je vytvořen profil.

3.3.2 UC2: Přihlásit se

Popis: uživatel se přihlásí jako host, nebo registrovaný.
Aktér: uživatel

Prekondice: uživatel má zobrazené příslušné přihlašovací okno.
Postkondice: uživatel je přihlášen.

Scénář:

1. Uživatel zvolí možnost přihlášení jako registrovaný.
   (a) Uživatel zvolí možnost přihlášení jako host.
      i. Bod 5.
2. Uživatel vyplní své přihlašovací údaje.
3. Uživatel potvrdí údaje tlačítkem.

4. Systém zkontroluje správnost údajů.
    (a) Uživatel zadal nesprávné údaje.
        i. Systém zobrazí upozornění.
        ii. Bod 2.
5. Uživatel je přihlášen.

3.3.3 UC3: Použití nástrojů k práci s tabulí

Popis: uživatel pracuje s tabulí pomocí různých nástrojů.
Aktér: uživatel - host, uživatel - registrovaný.

Prekondice: uživatel je přihlášen jako host nebo registrovaný.
Postkondice: uživatel změnil obsah tabule, nebo nastavil jiný nástroj pro další použití.

Scénář:

1. Uživatel zvolí nástroj z nabídky nástrojů.
2. Uživatel použije nástroj odpovídajícím způsobem.
3. Obsah tabule nebo nastavení nástrojů jsou změněny.

3.3.4 UC4: Uložení tabule

Popis: uživatel uloží obsah tabule na danou úložnou pozici.
Aktér: uživatel - registrovaný

Prekondice: uživatel je přihlášen jako registrovaný, uživatel má zobrazené okno uložených tabulí.
Postkondice: tabule je uložena v databázi aplikace.

Scénář:

1. Uživatel zvolí úložnou pozici.
2. Uživatel vyplní název tabule.
3. Uživatel potvrdí uložení tlačítkem.
4. Systém zkontroluje platnost názvu tabule.
    (a) Uživatel zadal neplatný název tabule.

i. Systém zobrazí upozornění.

ii. Bod 2.

5. Tabule je uložena do databáze aplikace.

3.3.5 UC5: Načtení tabule

Popis: uživatel načte tabuli z dané úložné pozice
Aktér: uživatel - registrovaný

Prekondice: uživatel je přihlášen jako registrovaný, uživatel má zobrazené okno uložených tabulí
Postkondice: obsah aktuální tabule je načten z databáze

Scénář:

1. Uživatel zvolí úložnou pozici s uloženou tabulí.
2. Uživatel potvrdí načtení tlačítkem.
3. Systém uživatele přemístí do nové skupiny.
4. Systém uživateli vygeneruje nový odkaz sdílení.
5. Tabule na dané úložné pozici přepíše obsah aktuální tabule.

Kapitola 4

Použité technologie

Dalším krokem ve vývoji aplikace bylo zvolit vhodné technologie. Ty byly vybrány podle předchozích zkušeností nabytých během studia i mimo něj.

4.1 HTML - HyperText Markup Language

Jedná se o základní stavební kámen webových stránek a aplikací. Definuje strukturu webového obsahu. Skládá se ze značek (tagů), které se píší do úhlových závorek a vytvářejí tak HTML elementy. Tyto elementy pak dále mohou mít různé atributy / vlastnosti. Řazení elementů za sebe poté vytváří výslednou webovou stránku. K propojení jednotlivých částí (stránek) aplikace využívá hypertextových odkazů. Základem k vytvoření moderního webu je také propojení HTML s dalšími technologiemi, jako jsou například CSS a JavaScript [8].

4.2 CSS - Cascading Style Sheets

Jazyk používaný k přizpůsobení grafické složky dokumentu napsaného pomocí jazyků HTML nebo XML a jazycích na něm založených. Mezi grafické prvky řízené CSS patří například pozice, barva, velikost, font daného elementu. CSS popisuje, jak by elementy dokumentu měly být vykresleny na obrazovku různých velikostí, papír a v dalších podobách. Může být vepsáno přímo do HTML dokumentu, vhodnější je však oddělit jej do samostatných souborů [9].

4.3 Less - Leaner Style Sheets

Less je zpětně kompatibilní rozšíření jazyka CSS, které přidává jen několik funkcí pro pohodlnější psaní CSS kódu [10].

4.4 JavaScript

JavaScript je multiplatformní, objektově orientovaný, událostmi řízený skriptovací jazyk. Je nejvíce známý jako skriptovací jazyk používaný k tvorbě webových stránek / aplikací, avšak je využíván i aplikacemi, které nevyužívají prostředí prohlížeče. Dovoluje dynamicky aktualizovat obsah webové stránky, ovládat její prvky, animovat je a mnoho dalších funkcí [11].

4.5 jQuery

jQuery není jazyk, jedná se o stručnou knihovnu JavaScriptu, jejíž cílem je zjednodušit práci se základním JavaScriptem. Obaluje mnoho často používaných víceřádkových funkcí JavaScriptu do metod, na jejichž zavolání stačí pouze jediný řádek. Zjednodušuje také práci s AJAX voláními a manipulaci s DOM [12].

4.6 Fabric.js

Jedná se o JavaScript knihovnu, která patří mezi základní stavební kameny aplikace. Značně zlepšuje práci s HTML canvasy (kreslícími plátny) tím, že umožňuje s nakreslenými čarami, vloženými obrázky a jinými elementy přidávanými na plátno pracovat jako s objekty. To poté umožňuje jejich pozdější upravování - přesunutí, změnu velikosti, roztažení, otočení a další. Dokáže také pracovat s vrstvami, čímž lze objekty přesouvat do popředí / pozadí, kreslení vzorem, animování objektů, funkci nekonečného plátna a mnoho dalších. Nabízí také funkci převedení celého plátna společně s objekty do formátu JSON, který může být uložen do databáze. Tato funkce je pro fungování aplikace kriticky důležitá [13].

4.7 C#

Jedná se o moderní, objektově orientovaný programovací jazyk vyvinutý zároveň s platformou .NET Framework. Je založen na jazycích jako jsou C, C++, Java a JavaScript, čímž umožňuje uživatelům ovládajícím tyto jazyky C# jednoduše využít [14].

4.8 ASP.NET Core

ASP.NET je webový framework, stručně řečeno se jedná o sadu knihoven, které umožňují tvorbu webových aplikací v jazyce C#. Knihovny obsahují hotová řešení mnoha základních problémů, které ve webových technologiích vyvstávají. To jsou např. bezpečnost, autentifikaci uživatele, práci s databází, správu formulářů a podobně.

Technologie je založena na architektuře klient-server. Aplikace v ASP.NET je tedy program, jehož výstupem je HTML stránka. ASP.NET běží na straně serveru.

ASP.NET Core se od ASP.NET MVC liší zejména tím, že na server nahráváme kromě naší aplikace i samotný framework ASP.NET, který již tedy nemusí být na serveru nainstalován. Svou aplikaci tedy můžeme nahrát kamkoli, i tam, kde není prostředí Windows nebo ASP.NET. Aby to nebylo moc jednoduché, tak kromě přístupu MVC přináší Core i další způsoby jak webové aplikace tvořit. ASP.NET Core je v současné době nejvyvíjenější, a proto se určitě vyplatí učit právě tuto technologii [15].

4.9 ASP.NET Core SignalR

SignalR je open-source knihovna, která do aplikace umožňuje a zjednodušuje přidat funkce, které probíhají v reálném čase.

SignalR představuje API pro vytváření volání ze serveru na klienty (RPC - Remote Procedure Calls), které tedy volají JavaScriptové funkce na straně klienta ze strany serveru postaveného na .NET Core. Automaticky spravuje připojení klientů, umožňuje zasílat informace všem klientům zároveň, zasílání informací pouze určitým klientům, dokáže zvládnout zvyšující se zatížení.

Ke komunikaci klientů a serveru využívá systém hubů. Tyto huby volají kód na straně klienta zasláním zprávy, která obsahuje název a parametry dané klientské metody. Klient poté vybere metodu, jijíž název se shoduje, zavolá ji a předá jí deserializovaná data parametrů [16].

4.10 Ostatní využité knihovny a moduly

4.10.1 Browser-image-compression.js

JavaScriptový modul, sloužící ke kompresi obrázkových souborů. Snižuje velikost a rozlišení nahrávaných obrázků, což má za následek snížení celkového objemu dat tabule [17].

4.10.2 Canvas-to-blob.js

Knihovna JavaScript umožňující převést objekt tabule Fabric.js na objekt Blob, který lze následně převést na obrázkový soubor [18].

4.10.3 FileSaver.js

JavaScript knihovna umožňující ukládat soubory na straně klienta. Využitá ve spojení s Canvas-to-blob pro uložení tabule Fabric.js jako obrázkového souboru [19].

4.10.4 Moment.js

Knihovna JavaScript pro převádění, úpravu a formátování dat [20].

4.10.5 Remove fblcid.js

Jednoduchá knihovna pro odstranění sledovacího parametru přidávaného do URL adresy po jejím zaslání přes síť Facebook [21].

4.11 Další možnosti

4.11.1 Socket.IO

Socket.IO je knihovna umožňující obousměrnou, na událostech založenou komunikaci mezi prohlížečem a serverem v reálném čase. Skládá se ze serveru Node.js a klientské JavaScript knihovny [22].

Socket.IO byl zpočátku součástí plánu na vytvoření aplikace, avšak díky nedostatku zkušeností se správou Node.js serveru dal přednost technologii SignalR.

4.11.2 oCanvas

Jedná se o JavaScript knihovnu, která má podobně jako použitý Fabric.js za cíl zjednodušit a vylepšit práci s HTML canvasy. Stejně jako Fabric.js, nepracuje přímo s pixely, ale s objekty [23].

Rozhodnutí nepoužít tuto technologii vzniklo po bližším prozkoumání její dokumentace, ze které bylo vyvozeno, že oCanvas není příliš zaměřen na volné kreslení.

4.11.3 Konva.js

Konva.js je další JavaScript knihovnou pro práci s HTML canvasy, která je dobrou alternativou k použitému Fabric.js. Aplikace by pravděpodobně mohla využívat tuto knihovnu místo knihovny Fabric.js, který byl oproti Konva.js zvolen díky vzhledově lepšímu režimu voného kreslení [24].

Zajisté existuje mnohem více různých technologií a knihoven, které by se daly při vývoji aplikace online sdílené tabule využít. Použité technologie se však při vývoji ukázaly jako dobře zvolené, jelikož během práce s nimi nevzniklo příliš mnoho problémů a komplikací.

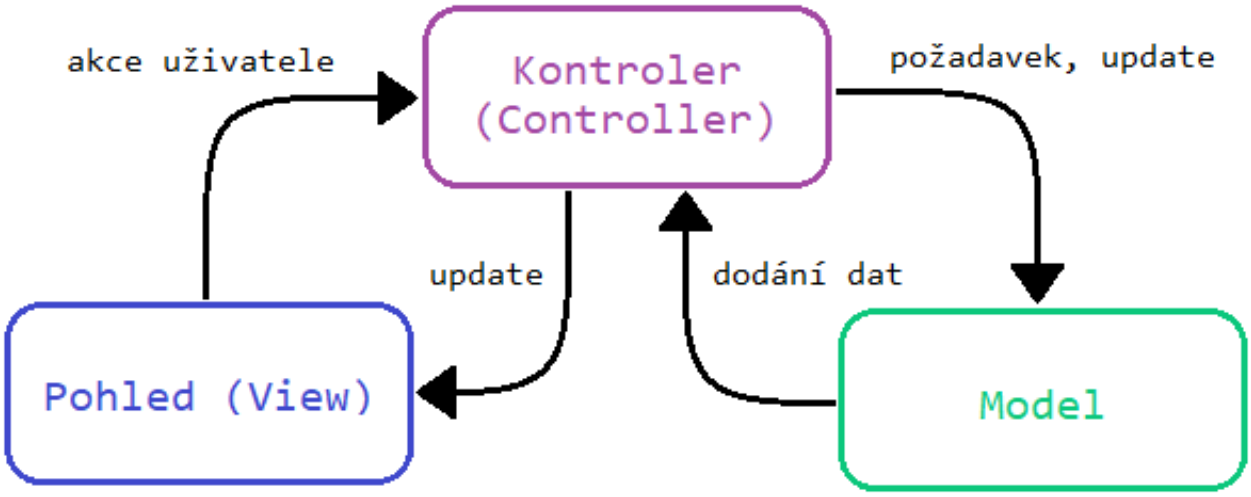
Od použitých technologií se také odvíjí název aplikace - Sigdraw, který vznikl spojením názvu SignalR a anglického slova draw (kreslit).

Kapitola 5

Architektura a struktura aplikace

5.1 ASP.NET Core MVC Architektura

Aplikace využívá architektury MVC, která ji rozděluje na tři hlavní komponenty - Model, View, Controller (model, pohled, kontroler). Toto rozdělení umožňuje úpravu jednotlivých částí bez zásadního porušení funkčnosti ostatních částí. Požadavky uživatele jsou směrovány na kontroler, který spolupracuje s modelem, aby došlo k jejich provedení nebo získání potřebných dat. Výsledek je poté většinou předán zpět pohledu, který jej vyžaduje pro zobrazení zpátky uživateli [25].



Obrázek 5.1: Společné fungování modelu, pohledu a kontroleru

5.2 Role komponentů ASP.NET Core MVC

5.2.1 Model

Model v MVC aplikaci představuje stav aplikace a jakoukoli business logiku nebo operace. Business logika by měla být uzavřena v modelu společně s veškerou implementační logikou pro uchování stavu aplikace. Zobrazení silného typu typicky využívají ViewModel typy navržené k zobrazení daných dat. Kontroler vytvoří a zaplní tyto instance ViewModelu daty z modelu [25].

5.2.2 Pohled

Pohledy jsou zodpovědny za prezentaci obsahu prostřednictvím uživatelského rozhraní. Používají Razor modul zobrazení k vložení kódu .NET do kódu HTML. V zobrazeních by mělo být logiky co nejméně a každá logika v nich by se měla vztahovat k prezentaci obsahu [25].

5.2.3 Kontroler

Kontrolery jsou komponenty, které zpracovávají interakci s uživatelem, pracují s modelem a nakonec určí pohled, který bude zobrazen. Zatímco pohled informace zobrazuje, kontroler zpracovává a odpovídá na vstup uživatele. V MVC architektuře je kontroler vstupním bodem, zodpovídá za výběr modelu, se kterým bude pracovat a který model bude zobrazen (z této funkce vychází jeho název - kontroler / řadič - řídí, jak aplikace zareaguje na určitý požadavek) [25].

5.3 Třídní diagram

Třídní diagram, znázorňující konkrétní třídy, jejich metody a vzájemné vazby, je díky své velikosti a pro lepší čitelnost uveden v příloze A.1 na straně 58.

5.4 Struktura aplikace

/wwwroot/
Obsahuje podsložky se soubory CSS, JavaScript, použitými fonty a přídavnými JavaScriptovými knihovnami.

/Controllers/
Složka, ve které jsou uloženy kontrolery aplikace.

/Core/
Uchovává třídy, které se starají o vnitřní logiku aplikace a obstarávají komunikaci s databází - systém přihlašování, registrace, vytváření tabulí, ukládání tabulí a další.

/Data/
Obsahuje třídy potřebné k prvotnímu nastavení databáze a jejímu zpřístupnění ostatním třídám.

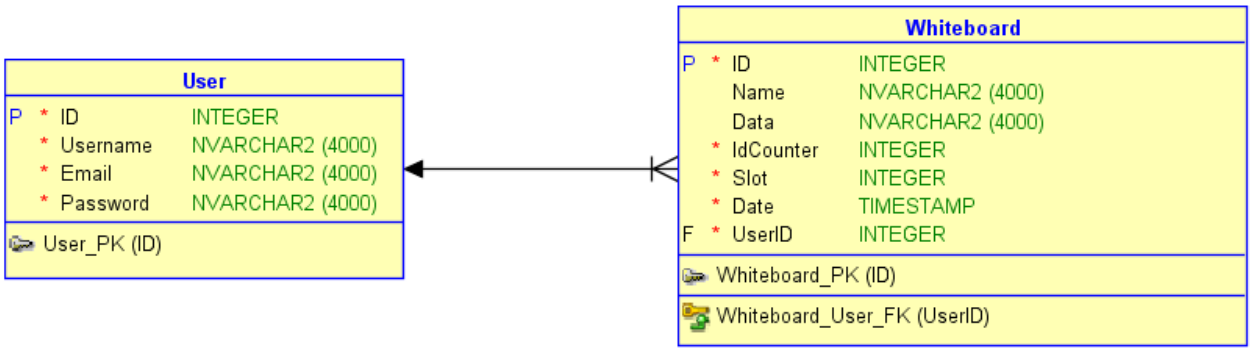
/Hubs/
V této složce jsou uloženy huby SignalR, které obstarávají komunikaci mezi uživateli.

/Models/
Uchovává modely aplikace.

/Views/
Uchovává pohledy aplikace.

5.5 Struktura databáze

Aplikace v současném stavu nevyžaduje složitou strukturu databáze a vystačí si jednoduchou databází obsahující pouze dvě tabulky ve vztahu 1:N.



Obrázek 5.2: ER diagram databáze aplikace

5.6 Lineární zápis databáze

Legenda: **Tabulka**, primární klíč, *cizí klíč*, atribut

User(ID, Username, Email, Password)
Whiteboard(ID, Name, Data, IdCounter, Slot, Date, *UserID*)

5.7 Datový model databáze

Tabulka User

Tato tabulka uchovává informace o uživateli. Aplikace uživatele nezatěžuje nutností zadávat žádné bližší údaje, jako je křestní jméno, příjmení, adresa a podobné, jelikož by se pro ně prozatím nenašlo využití.

Tabulka 5.1: Datový model tabulky **User**

Atribut	Dat. typ	Délka	Klíč	Null	Index	Význam
ID	INT		Primární	N	A	
Username	NVARCHAR2	Max		N		Jméno používané k přihlášení do aplikace.
Email	NVARCHAR2	Max		N		Email už., může být použit k identifikaci.
Password	NVARCHAR2	Max		N		Heslo k přihlášení uživatele.

Tabulka Whiteboard

V této tabulce jsou uloženy data jednotlivých tabulí. Každý uživatel má po registraci k dispozici tři úložné pozice tabulí, tedy tři záznamy v této tabulce.

Tabulka 5.2: Datový model tabulky **Whiteboard**

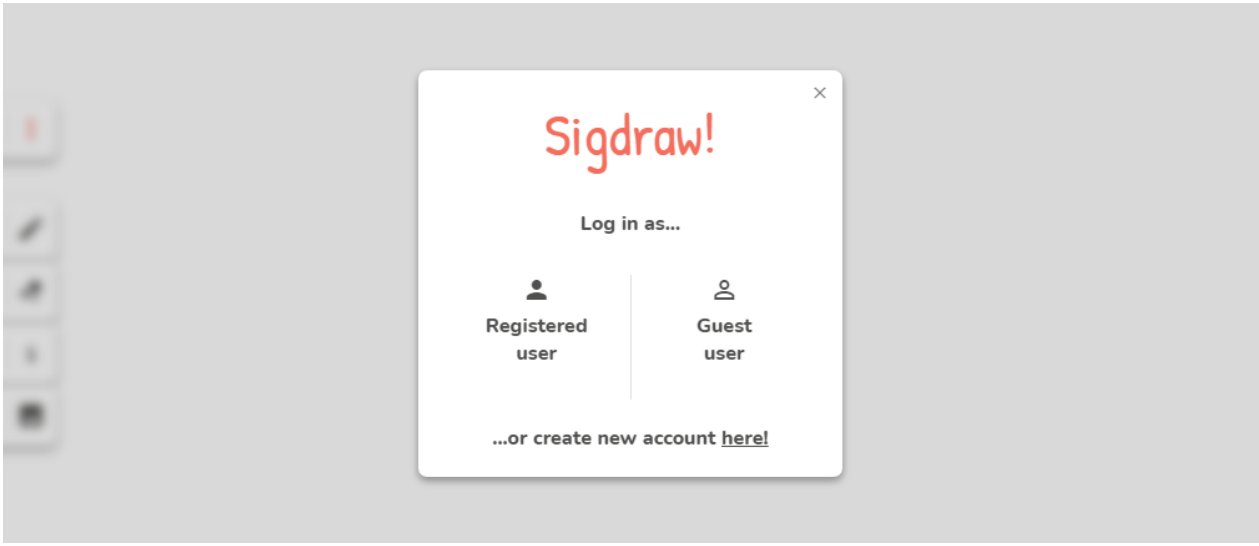
Atribut	Dat. typ	Délka	Klíč	Null	Index	Význam
ID	INT		Primární	N	A	
Name	NVARCHAR2	Max		A		Název tabule určený uživatelem.
Data	NVARCHAR2	Max		A		Data tabule ve formátu JSON.
IdCounter	INT			N		Udržuje ID následujícího objektu v tabuli.
Slot	INT	Max		N		Určuje úložnou pozici tabule.
Date	INT	Max		N		Datum uložení tabule.
UserID	INT		Cizí (User)	N		Odkazuje na vlastníka tabule.

Kapitola 6

Implementace funkcí a uživatelské rozhraní

6.1 Uživatelské rozhraní

Následujícím krokem při vývoji aplikace bylo vymyslet, jaký grafikcý styl bude mít. V dnešní době jsou aplikace často stylizovány minimalisticky (z testovaných aplikací například Miro nebo Whiteboard) a tato aplikace není výjimkou. Minimalisticky navržené rozhraní také podporuje celkové zaměření aplikace - jednoduchost a rychlost. Uživatel by měl být schopen se v aplikaci lehce orientovat a neměl by ztrácet čas navigováním mezi složitými nabídkami.



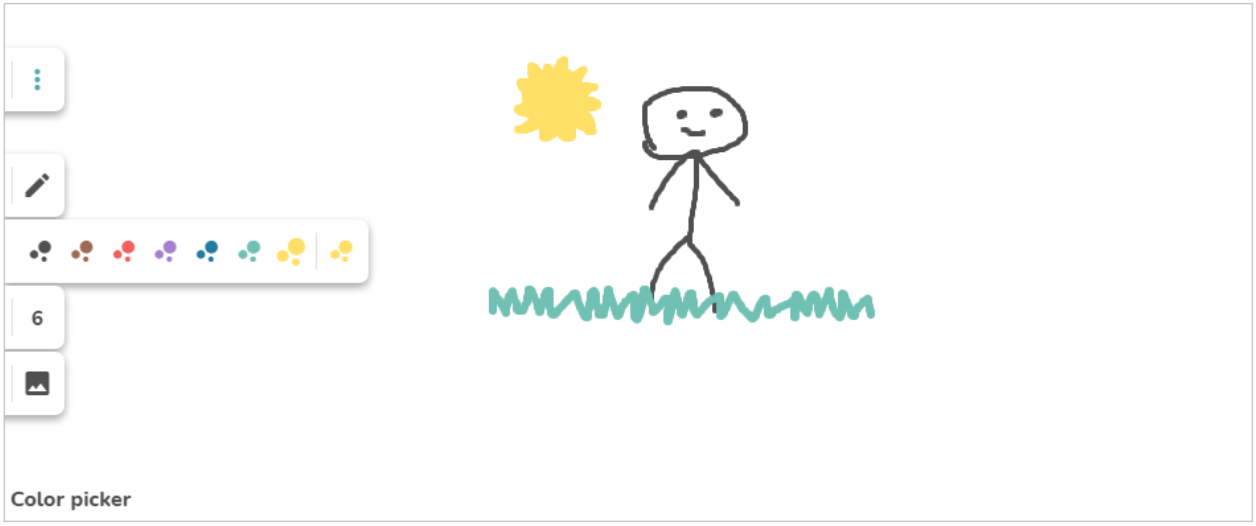
Obrázek 6.1: Přihlašovací okno zobrazené při prvním spuštění aplikace

Architektura aplikace je také navržena tak, že se všechno dění odehrává pouze na jedné HTML stránce, na které se pouze aktualizuje obsah. Tato funkce byla již od začátku hlavním bodem při vytváření uživatelského rozhraní. Aplikace takto nejen působí, ale i pracuje rychle. Této funkčnosti si lze nejvíce všimnout například při přihlašování nebo registraci, po kterých uživatel není přesměrován na jinou stránku, jak tomu bývá zvykem.

Hlavní ovládací prvek aplikace, nabídka nástrojů, se nachází na levé straně obrazovky. Je rozdělena do **následujících sekcí:**

1. **Hlavní nabídka** (obsahuje obecnější funkce pro ovládání aplikace)
2. **Nástroje kreslení**
3. **Nabídka barev**
4. **Nastavení velikosti čáry / písma**
5. **Nástroj vložení / vytvoření obrázku**

Tyto sekce jsou uživateli zčásti skryty a celé se zobrazí až poté, co na ně uživatel namíří kurzorem. To má za cíl co nejméně rušit uživatele od samotného kreslení a maximalizovat dostupnost kreslící plochy.



Obrázek 6.2: Kreslící plátno a nabídka nástrojů

Po přejetí kurzorem na určitou položku v nabídce nástrojů nebo po vykonání určité funkce se také v levém dolním rohu okna zobrazí stavový řádek. Ten slouží k informování uživatele o provedené funkci nebo zobrazení názvu zvoleného nástroje.

Bližší popis ovládacích prvků uživatelského rozhraní se nechází v manuálu přiloženém k této bakalářské práci.

6.2 Obecný popis jednotlivých skriptů

6.2.1 Draw.cshtml

Hlavní HTML soubor, který obsahuje všechny ovládací elementy aplikace. ASP.NET Core podporuje vkládání dynamického kódu C# do HTML souborů (díky tomu je přípona souboru CSHTML), avšak aplikace tuto funkci v současné podobě nevyužívá, jelikož o zobrazování dynamických dat se stará AJAX.

6.2.2 Draw.js

JavaScriptový soubor, který lze považovat za vstupní bod všech JavaScript funkcí. Má tři hlavní účely - obstarává ovládácí prvky stránky (práce s HTML elementy, změny stylování, vykreslování dynamických dat), informuje objekt třídy **DrawClient** o požadavcích zaslaných ze SignalR hubu a informuje objekt třídy **Whiteboard** o událostech, které vznikají nad samotnou tabulí (nakreslení čáry, změna barvy a další).

6.2.3 DrawClient.js

Skript tvořící stejnojmennou třídu. Tato třída má za úkol zpracovávat připojení uživatele k tabuli, zpracovávat informace o akcích uživatele provedených na tabuli ze souboru **Draw.js** a tyto informace předávat buď SignalR hubu nebo objektu **Whiteboard**. Stará se také o volání AJAX funkcí, ve kterých zasílá požadavky příslušnému kontroleru. Ve zkratce by se tedy dal označit jako prostředník mezi kódem JavaScript a C#.

6.2.4 Whiteboard.js

Tento skript tvoří třídu **Whiteboard**, tedy třídu samotné tabule. Třída obsahuje objekt plátna Fabric.js, a zaobírá se metodami, které se volají nad tímto objektem - vykreslení čáry, písma, obrázků, transformace objektů a dalších.

6.2.5 DrawHub.cs

SignalR hub zaměřující se na informování ostatních klientů o změnách provedených na tabuli. Očekává volání svých metod ze třídy **DrawClient**. Pokud toto volání nastane, informuje ostatní uživatele (klienty) o tom, která metoda byla volána a předá jí potřebné parametry, opět získané od třídy **DrawClient**.

6.2.6 AuthController.cs

Jeho hlavním účelem je předávat data mezi **DrawClient.js** a **AuthService.cs**.

6.2.7 WhiteboardController.cs

Podobně jako **AuthController.cs**, hlavním účelem tohoto skriptu je připravovat a předávat data mezi **DrawClient.js** a **WhiteboardService.cs**.

6.2.8 AuthService.cs

Třída **AuthService** má na starosti veškerou správu uživatelů - registrace, přihlašování, určení, zda je uživatel přihlášen v roli registrovaného nebo neregistrovaného. Kontroluje také, zda uživatel zadává validní data při různých událostech.

6.2.9 WhiteboardService.cs

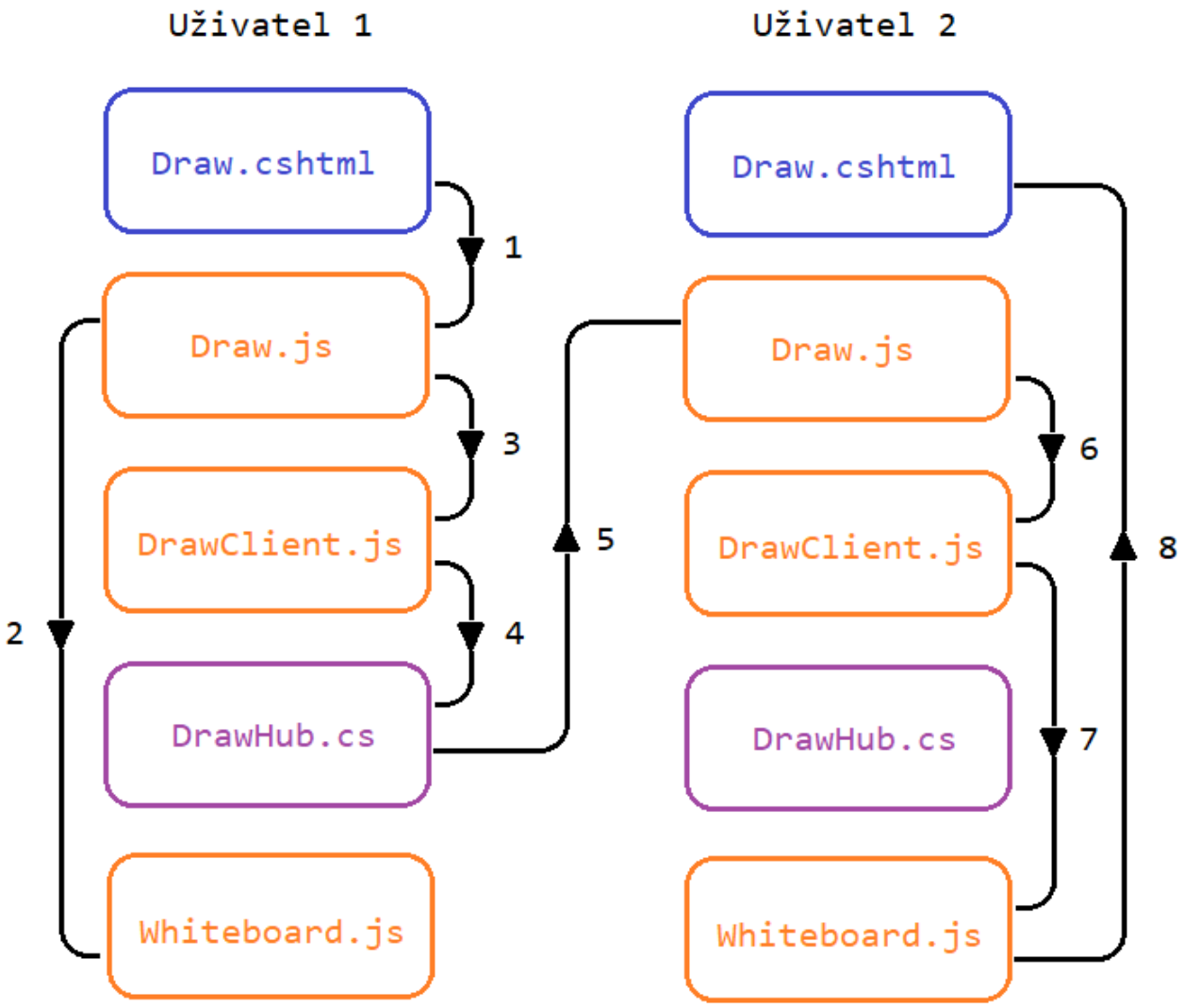
WhiteboardService je třída, která má na starost veškeré funkce tabule, které pracují s databází. Mezi takové se řadí například načítání informací o tabuli, načítání a ukládání tabulí.

6.3 Postup volání skriptů

Volání skriptů při provedených uživatelských akcích je možné rozdělit na dva postupy.

6.3.1 První postup

První postup je proveden, pokud uživatel provedl akci, která nezahrnuje komunikaci s databází, ale využívá SignalR hubu ke komunikaci s ostatními uživateli (nakrelsení čáry, transformace objektu nebo jiné).



Obrázek 6.3: První postup volání skriptů

1. Uživatel provede akci na zobrazené HTML stránce (**Draw.cshtml**). Tato akce je odchycena v souboru **Draw.js** (nakreslení čáry, posun objektu nebo jiné).

2. O akci je v souboru **Draw.js** informován objekt **Whiteboard**, čímž je zavolána příslušná metoda ve **Whiteboard.js**.

3. O akci je také informován objekt **DrawClient**, je tedy zavolána příslušná metoda v **DrawClient.js**.

4. Metoda v **DrawClient.js** zpracuje vstupní parametry předané z **Draw.js** a předá je SignalR hubu v souboru **DrawHub.cs**, který zavolá požadovanou metodu.

5. **DrawHub** informuje všechny potřebné klienty o tom, kterou metodu mají zavolat, a předá jim potřebné parametry.

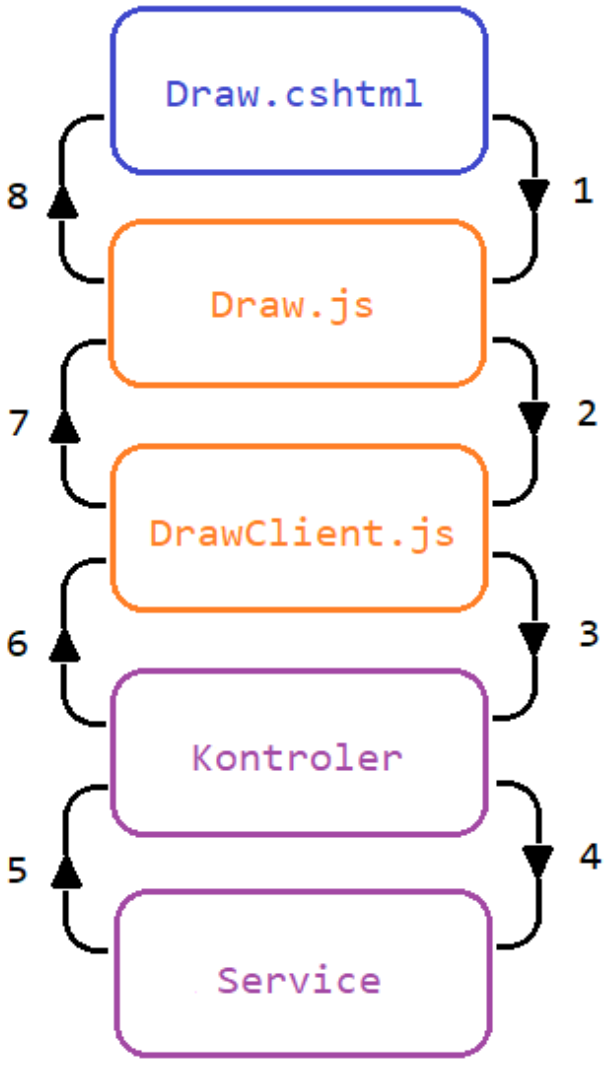
6. Volání ze souboru **DrawHub.cs** jsou u daných klientů zachyceny objektem třídy **DrawClient** v souboru **Draw.js**.

7. Může nastat potřeba provést dodatečné metody v **DrawClient**. Po provedení těchto metod se zavolají metody ze souboru **Whiteboard.js**, čímž se akce provedené původním uživatelem projeví u všech potřebných ostatních uživatelů.

8. Akce se vizuálně projeví aktualizovanou HTML stránkou **Draw.cshtml**.

6.3.2 Druhý postup

Druhý postup volání skriptů nastane, pokud uživatel provedl akci, která komunikuje s databází (registrace, přihlášení, uložení a načtení tabule a jiné).



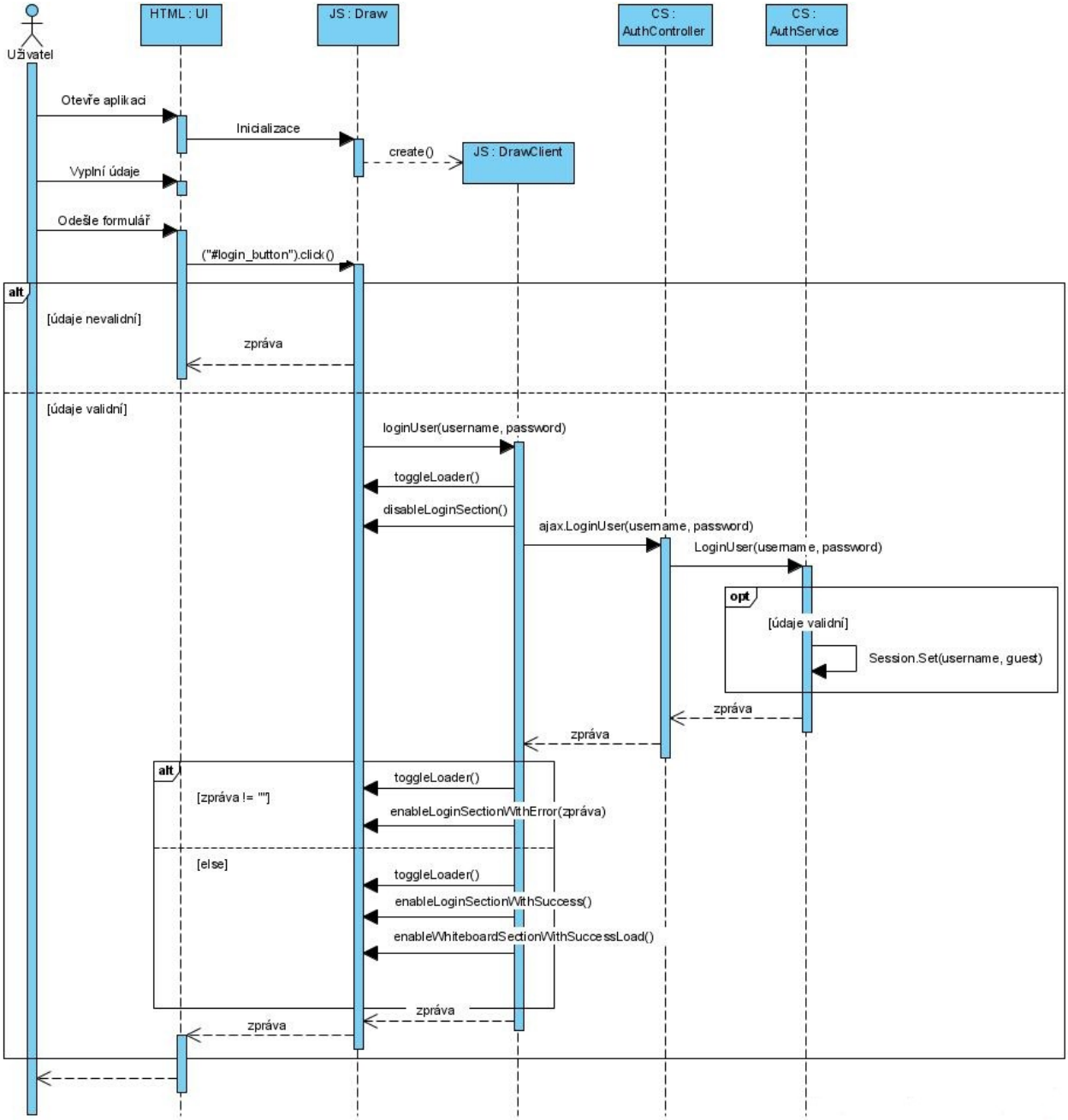
Obrázek 6.4: Druhý postup volání skriptů

1. Uživatel provede akci na zobrazené HTML stránce (**Draw.cshtml**). Tato akce je odchycena v souboru **Draw.js** (přihlášení, registrace, uložení a načtení tabule a jiné).
2. O akci je v souboru **Draw.js** informován objekt **DrawClient**, čímž je zavolána příslušná metoda v **DrawClient.js**, které jsou předány uživatelem vyplněné parametry.
3. Metoda je tvořena AJAX voláním příslušného kontroleru (**AuthController.cs**, **WhiteboardController.cs**).
4. Kontroler předá parametry dané Service třídě (**AuthService.cs**, **WhiteboardService.cs**), která provede operace nad databází.
5. Service třída podá výstup operace zpět kontroleru.
6. Kontroler tento výstup podobně předá zpět objektu **DrawClient**.
7. **DrawClient** poté podle výstupu určí, pomocí jaké metody v **Draw.js** bude výstup zobrazen uživateli.
8. Výstup je uživateli zobrazen na aktualizované HTML stránce (**Draw.cshtml**).

6.4 Bližší popis skriptů

Kód aplikace je pro co nejlepší pochopení popsán na třech scénářích. První scénář popisuje přihlášení uživatele, druhý scénář přidání objektu na plátno tabule a třetí scénář popisuje proces uložení tabule do databáze. U každé části kódu je v závorce uveden soubor, ve kterém se daný kód nachazí.

6.4.1 První scénář - přihlášení uživatele



Obrázek 6.5: Sekvenční diagram - přihlášení registrovaného uživatele

Uživatel může do aplikace přistoupit dvěma zůsoby - sám navštíví čistou webovou adresu aplikace, nebo aplikaci otevře přes odkaz sdílení, který mu byl zaslán jiným uživatelem. Tento odkaz určuje, se kterými ostatními uživateli sdílí daný uživatel tabuli. Nejdříve je tedy nutno vyřešit URL adresu. Pokud je stránka navštívena bez odkazu sdílení, je vygenerován nový řetězec náhodných znaků, který je následně k čisté URL adrese připojen. Pokud již URL tento řetězec obsahuje, nedojde ke generování nového.

```
createNewLobby() {
      var code = this.randomString(12);
      window.history.replaceState(null, null, "?whiteboard=" + code);
      this.removeFromGroup(this.group, true, null, true);
      this.group = window.location.href;
      this.addToGroup(this.group, true, null, true);
  }
```

Kód 6.1: Úprava URL adresy třídou DrawClient **(DrawClient.js)**

Poté je uživatel přidán do skupiny, která nese stejný název jako je aktuální URL adresa. Přidáním uživatelů do skupin dojde k oddělení jejich komunikace - uživatelé komunikují pouze s těmi uživateli, kteří jsou ve stejné skupině. Skupiny se řadí do funkcí SignalR hubu, je tedy nutné ho zavolat.

```
addToGroup(group) {
      this.connection.invoke("AddToGroup", group).catch(function (err) {
          return console.error(err.toString());
      });
  }
```

Kód 6.2: Zavolání SignalR hubu pro přidání uživatele do skupiny **(DrawClient.js)**

Connection je zde atribut objektu **DrawClient**, pomocí kterého je navázáno spojení se SignalR hubem. Na připojení je vyvolána funkce specifikována názvem *AddToGroup* a je jí předán parametr *group* (URL adresa vytvořena dříve). SignalR hub se v následujícím kroku postará o přidání uživatele do skupiny.

```
public async Task AddToGroup(string group)
{
   await Groups.AddToGroupAsync(Context.ConnectionId, group);
}
```

Kód 6.3: Přidání uživatele do skupiny v SingalR hubu **(DrawHub.cs)**

Uživatelova komunikace je tedy úspěšně oddělena od ostatních skupin a je mu tedy dále zobrazeno vyskakovací okno, které mu dá na výběr ze dvou způsobů přihlášení - registrovaný, nebo host. V tomto okně je obsažena také možnost se zaregistrovat, ta zde však nebude podrobněji popsána. Pokud uživatel zvolí možnost přihlášení jako host, je zavolána následující funkce:

```
loginGuest() {
     $.ajax({
         type: 'post',
         url: '/Auth/LoginGuest',
         success: function (result) {
             disableWhiteboardSection();
         }
     });
 }
```

Kód 6.4: AJAX funkce volající AuthController **(DrawClient.js)**

Podle klíčového slova ajax je jasné, že se jedná o AJAX funkci. Pomocí parametru *url* je určeno, že bude zaslán požadavek na kontroler **AuthController** a v něm zavolána funkce *LoginGuest*. Za parametrem *url* se nachází parametr *success*. Tento parametr označuje, co se stane po provedení volané funkce a jaké návratové hodnoty z ní přijdou. V tomto případě dojde ke znemožnění přístupu k uloženým tabulím, neboť tato funkce je dostupná pouze pro registrované uživatele.

```
public JsonResult LoginGuest()
{
   var result = _auth_service.LoginGuest();
   return Json(result);
}
```

Kód 6.5: Volání AuthService uvnitř kontroleru AuthController **(AuthController.cs)**

Kontroler **AuthController** poté předá data třídě **AuthService** zavoláním její funkce *LoginGuest*, která provede přihlášení uživatele jako hosta. Návratová hodnota funkce *LoginGuest* bude později vrácena zpět třídě **DrawClient** ve formě JSON řetězce.

```
public string LoginGuest()
{
    if (_http_context_accessor.HttpContext.Session.GetInt32("guest") == null ||
        _http_context_accessor.HttpContext.Session.GetInt32("guest") == 0)
    {
        string guestname = "Guest-" + RandomString(6);
        _http_context_accessor.HttpContext.Session.SetString("username", guestname
            );
        _http_context_accessor.HttpContext.Session.SetInt32("guest", 1);
        return "";
    }

    return "Already logged in";
}
```

Kód 6.6: Přihlášení uživatele jako hosta **(AuthService.cs)**

Funkce *LoginGuest* zkontroluje, zda již uživatel není přihlášen jako host. V případě, že opravdu není hostem, vygeneruje nové jméno a nastaví potřebné Session parametry (parametry relace), díky kterým se uživatel stává přihlášeným jako host. Hlášení o úspěchu či neúspěchu je přes **AuthController** vráceno zpět třídě **DrawClient**, která následně provede potřebné metody.

V případě, že uživatel zvolí možnost přihlášení jako registrovaný, je mu zobrazen přihlašovací formulář. Po potvrzení formuláře jsou zadané údaje zkontrolovány na straně uživatele pomocí JavaScript funkce, která případně zobrazí hlášení o jejich neplatnosti.

```
$('#login_button').click(function () {
    if (!$('#login_form')[0].reportValidity()) {
        return false;
    }
    var username = $('#login_username').val();
    var password = $('#login_password').val();
    if (username.length < 4 || username.length > 16) {
        $('#login_result').text('Invalid username');
        return false;
    }
    ...
    draw_client.loginUser(username, password);
});
```

Kód 6.7: Kontrola přihlašovacích údajů na straně uživatele **(Draw.js)**

Na konci této funkce je zavolán objekt třídy **DrawClient**, přesněji jeho funkce *LoginUser*.

```
loginUser(username, password) {
     toggleLoader();
     disableLoginSection();
     $.ajax({
         type: 'post',
         data: {username: username, password: password},
         url: '/Auth/LoginUser',
         success: function (result) {
             if (result != "") {
                 toggleLoader();
                 enableLoginSectionWithError(result);
             }
             else {
                 toggleLoader();
                 enableLoginSectionWithSuccess();
                 enableWhiteboardSectionWithSuccessLoad();
             }
         }
     });
 }
```

Kód 6.8: AJAX volání kontroleru AuthController s parametry **(DrawClient.js)**

Funkce je podobná dříve popsané AJAX funkci *loginGuest*, rozdílem je však to, že jsou kontroleru **AuthController** předávány parametry, se kterými bude pracovat. Tyto parametry tvoří přihlašovací údaje dříve zkontrolované v souboru **Draw.js**.

```
public JsonResult LoginUser(string username, string password)
{
   string result = _auth_service.LoginUser(username, password);
   return Json(result);
}
```

Kód 6.9: Předání parametrů z AuthController třídě AuthService **(AuthController.cs)**

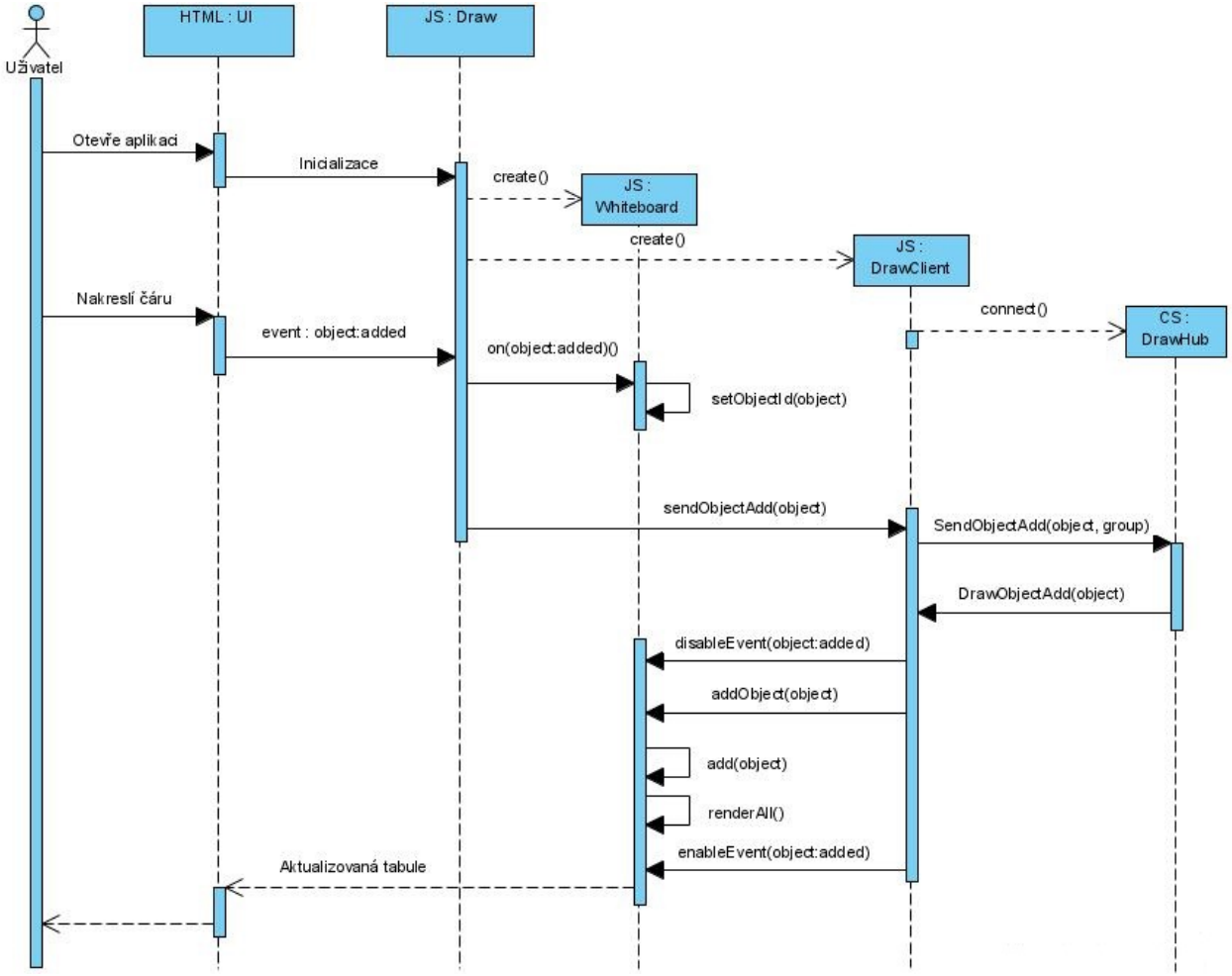
Následně je zavolána funkce *LoginUser* třídy **AuthService** s potřebnými parametry.

```
public string LoginUser(string username, string password)
    {
        if (username.Length < 4 || username.Length > 16)
        {
            return "Invalid username";
        }
     ...
     var user = GetUserByUsername(username);
     ...

        var passwordVerificationResult = new PasswordHasher<object?>().
            VerifyHashedPassword(null, user.Password, password);

        switch (passwordVerificationResult)
        {
            case PasswordVerificationResult.Failed:
                return "Invalid username or password";

            case PasswordVerificationResult.Success:
                _http_context_accessor.HttpContext.Session.SetString("username",
                    user.Username);
                _http_context_accessor.HttpContext.Session.SetInt32("guest", 0);
                return "";

            default:
                return "Invalid username or password";
        }
    }
```

Kód 6.10: Kontrola údajů a přihlášení registrovaného uživatele **(AuthService.cs)**

Uživatelem zadané přihlašovací údaje jsou opět zkontrolovány, tentokrát na straně serveru. Poté je provedeno šifrování zadaného hesla pomocí vnitřní metody ASP.NET Core, hesla uživatelů jsou totiž v databázi uložena v této šifrované podobě. Pokud heslo odpovídá jménu uživatele a shoduje se s jeho heslem uloženým v databázi, je uživatel přihlášen jako registrovaný pomocí nastavení parametru Session. Hlášení o úspěchu či neúspěchu je přes kontroler **AuthController** předáno zpět třídě **DrawClient**, která se společně s **Draw.js** postará o zobrazení výstupu uživateli.

6.4.2 Druhý scénář - přidání objektu na tabuli



Obrázek 6.6: Sekvenční diagram - přidání objektu na tabuli

Při přidání objektu na tabuli (nakreslení čáry a dalších) nebo i jakékoli jiné práci s plátnem tabule (transformace objektu) je třeba využít SignalR hubu, nikoli AJAX volání kontroleru a následné práce s databází. Je pouze potřeba informovat ostatní uživatele ve skupině o provedené akci. Akce (událost) je odchycena následující funkcí:

```
whiteboard.canvas.on('object:added', function (e) {
  whiteboard.setObjectId(e.target);
  draw_client.sendObjectAdd(e.target);
});
```

Kód 6.11: Odchycení události přidání objektu na tabuli **(Draw.js)**

Whiteboard je objektem vytvořeným na začátku skriptu **Draw.js**. K tomuto objektu má přístup také objekt třídy **DrawClient**. Objekt přidaný na plátno tabule je při tomto odchycení označován jako *e.target* (čára, obrázek, text). Fabric.js jako takový nijak neudržuje přehled o tom, který objekt je který, je tedy potřeba přidávanému objektu nastavit identifikátor (metoda *setObjectId*). Po přidání identifikátoru bude informace o přidání objektu společně s daným objektem odeslána dalším klientům ve skupině (metoda *sendObjectAdd*).

```
setObjectId(object) {
      object['id'] = this.id_counter;
      this.id_counter += 1;
  }
```

Kód 6.12: Přidání objektu na tabuli **(Whiteboard.js)**

Nastavení identifikátoru nově přidaného objektu je nezbytně důležité pro umožnění jeho pozdějšího upravování. Třída **Whiteboard** si udržuje počítadlo těchto identifikátorů a při každém přidaném objektu toto počítadlo navýší. Po přiřazení identifikátoru je zavolána metoda *sendObjectAdd* třídy **DrawClient**.

```
sendObjectAdd(object) {
      this.connection.invoke("SendObjectAdd", object, this.group).catch(function
           (err) {
          return console.error(err.toString());
      });
  }
```

Kód 6.13: Volání SignalR hubu třídou DrawClient **(DrawClient.js)**

Jelikož si třída **DrawClient** udržuje aktuální připojení k SignalR hubu, může jednoduše zavolat jeho metody. V tomto případě zavolá metodu *SendObjectAdd* a předá jí samotný objekt k přidání a specifikaci skupiny, jejímž členům bude objekt zaslán.

```
public async Task SendObjectAdd(object json, string group)
{
   await Clients.OthersInGroup(group).SendAsync("DrawObjectAdd", json);
}
```

Kód 6.14: Volání klientských metod SignalR hubem **(DrawHub.cs)**

SignalR vyšle požadavek zavolání metody *DrawObjectAdd* všem klientům připojeným do dané skupiny. Přidávaný objekt je automaticky převeden na JSON formát.

```
draw_client.connection.on("DrawObjectAdd", function (json) {
   draw_client.drawObjectAdd(json);
});
```

Kód 6.15: Zachycení volání funkce ze SignalR hubu **(Draw.js)**

Volání SignalR hubu je u všech potřebných klientů zachyceno objektem **DrawClient**, který těmto voláním naslouchá. Dále je zavolána metoda této třídy, která má jako parametr objekt obdržený ze SignalR hubu. Tento objekt má být přidán na tabule všech daných klientů.

```
drawObjectAdd(json) {
     this.whiteboard.disableEvent("object:added");
     this.whiteboard.addObject(json);
     this.whiteboard.enableEvent('object:added');
 }
```

Kód 6.16: Příprava tabule pro přidání příchozího objektu **(DrawClient.js)**

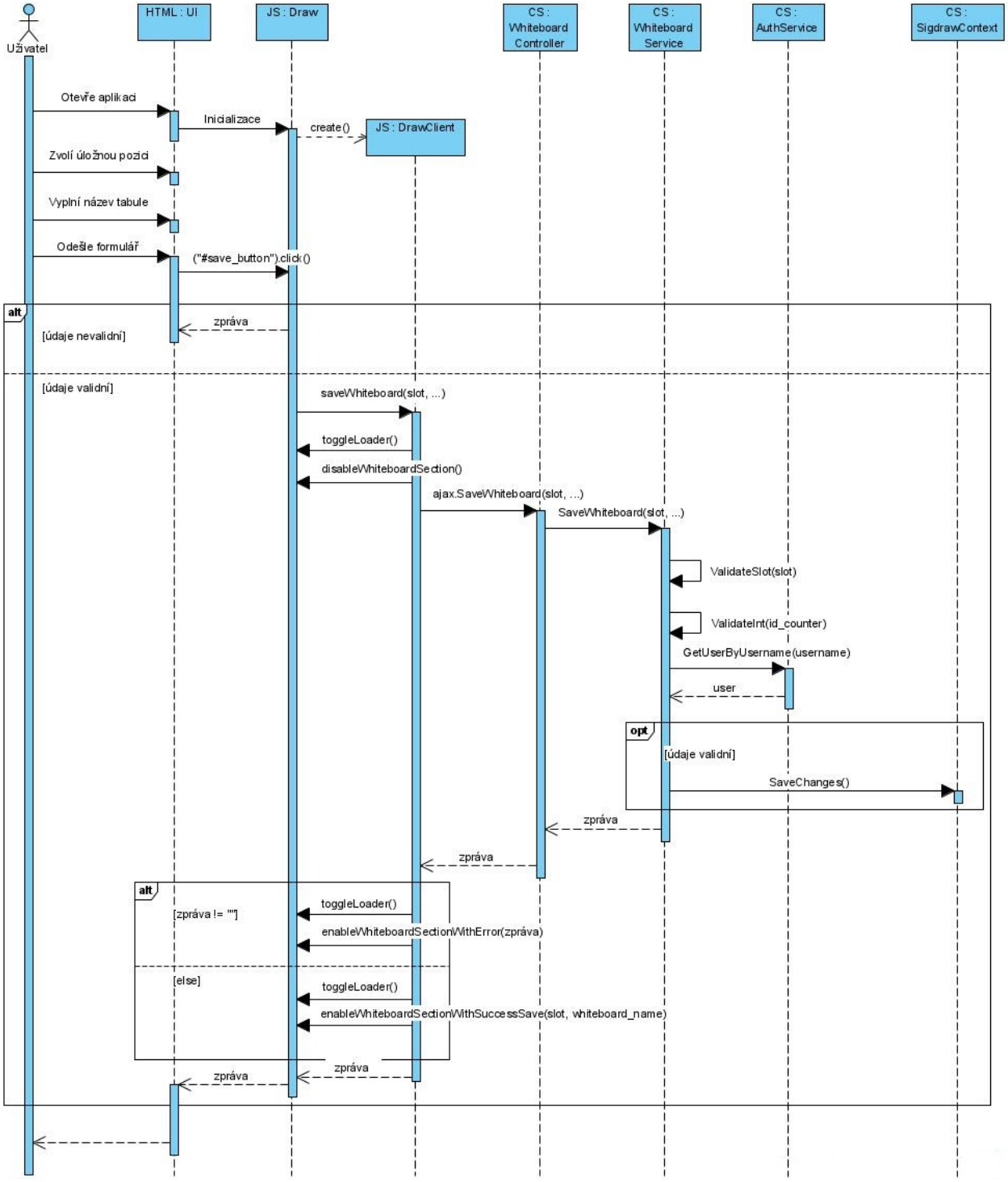
Před přidáním objektu na tabule klientů je nutné zamezit vyvolání události přidání objektu, neboť jak již bylo zmíněno, vyvolání této události je zachyceno a vede k zasílání přidaného objektu ostatním uživatelům. Je tedy potřeba předejít cyklickému volání SignalR hubu a přidávání objektu na všech tabulích připojených klientů.

```
addObject(json) {
      this.setObjectId(json);

      var temp = this.canvas;

      fabric.util.enlivenObjects([json], function (enlivenedObjects) {
          temp.add(enlivenedObjects[0]);
          temp.renderAll();
      });
      this.canvas = temp;
  }
```

Kód 6.17: Přidání příchozího objektu na tabuli **(Whiteboard.js)**

Po přidělení identifikátoru příchozímu objektu dojde na jeho samotné přidání na tabuli. Metoda *enlivenObjects* vytvoří instanci objektu z jeho objektové reprezentace ve formě JSON, načež je objekt přidán na plátno, které je překresleno, aby došlo k zobrazení tohoto nového objektu.

6.4.3 Třetí scénář - uložení tabule do databáze



Obrázek 6.7: Sekvenční diagram - uložení tabule do databáze

Tento scénář probíhá podobným způsobem jako první scénář, není tedy nutné popisovat všechny části kódu. Popsány jsou jen části lišící se od prvního scénáře.

Pokud je uživatel přihlášen jako registrovaný, má možnost si své kresby ukládat do databáze, přičemž má k dispozici tři úložné pozice. Tyto pozice jsou uživateli přiděleny již při registraci.

```
$('#dialog_whiteboards_save').click(function () {
  if (!$('#whiteboard_form')[0].reportValidity()) {
    return false;
  }

  var whiteboard_name = $('#whiteboard_new_name').val();

  if (whiteboard_name == "") {
    $('#whiteboard_result').text('Invalid whiteboard name');
    return false;
  }

  if (whiteboard_name.Length > 16) {
    $('#whiteboard_result').text('Whiteboard name must be between 1 and 16
        characters');
    return false;
  }

  var slot = $('.selected_whiteboard').attr('id');
  var whiteboard_data = JSON.stringify(whiteboard.canvas.toJSON(['id']));
  var id_counter = whiteboard.id_counter;
  draw_client.saveWhiteboard(slot, whiteboard_data, whiteboard_name, id_counter)
      ;
});
```

Kód 6.18: Příprava dat ukládané tabule **(Draw.js)**

Po vybrání úložné pozice a kliknutí na tlačítko uložení je zkontrolována správnost názvu tabule vyplněného uživatelem. Celá tabule je dále převedena na JSON řetězec, aby mohla být uložena do databáze. Je také nutné uložit její počítadlo identifikátorů přidaných objektů, aby nedošlo k jejich přepisování při budoucí práci s tabulí. Řetězec reprezentující tabuli a počítadlo identifikátorů objektů jsou následně zaslány objektu **DrawClient**, který pomocí AJAX volání kontroleru **WhiteboardController** předá všechny tyto data třídě **WhiteboardService**, která se postará o jejich uložení.

```
public string SaveWhiteboard(string slot, string whiteboard_data, string
    whiteboard_name, string id_counter)
    {
        if (whiteboard_name == "" || whiteboard_name == "-")
        {
            return "Invalid whiteboard name";
        }

        ...

        Whiteboard whiteboard = _context.whiteboards.Where<Whiteboard>(w => w.
            UserID == user.ID).Where<Whiteboard>(w => w.Slot == validated_slot)
            .FirstOrDefault();

        if (whiteboard == null)
        {
            return "An error occured while saving the whiteboard";
        }

        whiteboard.Name = whiteboard_name;
        whiteboard.Data = whiteboard_data;
        whiteboard.IdCounter = validated_id_counter;
        whiteboard.Date = DateTime.Now;

        _context.SaveChanges();

        return "";
    }
```

Kód 6.19: Uložení tabule do databáze **(WhiteboardService.cs)**

Po kontrole správnosti vstupních parametrů následuje samotné uložení tabule do databáze. Nejdříve je ale potřeba zkontrolovat, zda je daná úložná pozice pro tabuli uživatele dostupná. Pokud tomu tak je, objektu tabule jsou přiřazeny potřebné atributy a je uložen do databáze pomocí metody *SaveChanges*, která je součástí ASP.NET Core.

Kapitola 7

Zhodnocení a závěr

Výsledkem této bakalářské práce je webová aplikace, která uživatelům umožňuje kreslit na plátno sdílené online v reálném čase. Tvorba takového nástroje byla umožněna správně zvolenými a použitými technologiemi, díky kterým při vývoji nevznikaly závažnější komplikace.

Dalo by se říci, že práce na takovýchto webových aplikacích nikdy nekončí - stále je třeba aplikaci vylepšovat a přidávat nové funkce, aby byla schopná konkurovat ostatním aplikacím na trhu, které jsou oproti této aplikaci vytvářeny velkými vývojovými týmy.

I přesto, že při vytváření aplikace nevznikaly závažné komplikace, objevilo se pár méně závažných, které nijak výrazně nezpomalily její vývoj.

První komplikací byla nemožnost ukládání tabule do databáze, která byla zapříčiněna špatně zvolenou metodou převodu tabule do textového formátu. Tato metoda využívala algoritmů, které nezvládaly zpracovávat velké množství textu a docházelo k zaplnění dostupné paměti. Určení místa závady a řešení tohoto problému zabralo při vývoji přibližně dva dny, přestože se na pohled může zdát, že jde o drobnou chybu.

Druhou výraznější komplikací bylo zjištění, že SignalR není schopen pracovat se Session (relačními) proměnnými prohlížeče. Tuto skutečnost však řeší použití AJAX, který umožňuje volat kontroler a třídy, které již k Session proměnným přístup mají. Je ale možné, že takto dojde k dalším komplikacím při budoucím rozšiřování aplikace. Aplikace je sestrojena takovým způsobem, že toto následné rozšiřování umožňuje.

Mezi funkce, které by mohly být v budoucnu přidány, patří například historie akcí uživatelů, kteří takto by mohli navrátit změny provedené na tabuli, nebo přidání dalších nástrojů kreslení - kreslení geometrických obrazců, kreslení vzorem, vyplnění oblasti barvou a mnoha dalších.

Literatura

1. ZUNENSHINE, Michael. *13 Best Online Whiteboard Collaboration & Teaching Tools 2021* [online]. 2020-09-22 [cit. 2021-04-06]. Dostupné z: `https://crm.org/news/best-online-whiteboard`.
2. *Miro | Free Online Collaborative Whiteboard Platform* [online] [cit. 2021-04-06]. Dostupné z: `https://miro.com/`.
3. *Stormboard* [online] [cit. 2021-04-06]. Dostupné z: `https://stormboard.com/`.
4. *Limnu - The Online Whiteboard You've Been Looking For.* [Online] [cit. 2021-04-06]. Dostupné z: `https://limnu.com/`.
5. *Simple Online Whiteboard - Whiteboard Fox* [online] [cit. 2021-04-06]. Dostupné z: `https://r7.whiteboardfox.com/`.
6. *Aplikace digitální online tabule – Microsoft Whiteboard* [online] [cit. 2021-04-06]. Dostupné z: `https://www.microsoft.com/cs-cz/microsoft-365/microsoft-whiteboard/digital-whiteboard-app`.
7. *Witeboard* [online] [cit. 2021-04-06]. Dostupné z: `http://www.witeboard.com`.
8. *HTML: HyperText Markup Language | MDN* [online] [cit. 2021-04-06]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/HTML`.
9. *CSS: Cascading Style Sheets | MDN* [online] [cit. 2021-04-06]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/CSS`.
10. *Getting Started | Less.Js* [online] [cit. 2021-04-06]. Dostupné z: `http://lesscss.org/#overview`.
11. *JavaScript | MDN* [online] [cit. 2021-04-06]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/JavaScript`.
12. *jQuery Introduction* [online] [cit. 2021-04-06]. Dostupné z: `https://www.w3schools.com/jquery/jquery_intro.asp`.
13. *Introduction to Fabric.Js. Part 1. — Fabric.Js Javascript Canvas Library* [online] [cit. 2021-04-06]. Dostupné z: `http://fabricjs.com/fabric-intro-part-1#why_fabric`.

14. BILLWAGNER. *Prohlídka Průvodce C# – C#* [online] [cit. 2021-04-06]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/csharp/tour-of-csharp/.

15. ČÁPKA, David. *Lekce 1 - Úvod do ASP.NET* [online] [cit. 2021-04-06]. Dostupné z: https://www.itnetwork.cz/tutorial-uvod-do-asp-dot-net.

16. BRADYGASTER. *Úvod do ASP.NET Core SignalR* [online] [cit. 2021-04-06]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/signalr/introduction.

17. CHAN, Donald. *Donaldcwl/Browser-Image-Compression* [online]. 2021-03-31 [cit. 2021-04-06]. Dostupné z: https://github.com/Donaldcwl/browser-image-compression.

18. TSCHAN, Sebastian. *Blueimp/JavaScript-Canvas-to-Blob* [online]. 2021-03-28 [cit. 2021-04-06]. Dostupné z: https://github.com/blueimp/JavaScript-Canvas-to-Blob.

19. GREY, Eli. *Eligrey/FileSaver.Js* [online]. 2021-04-06 [cit. 2021-04-06]. Dostupné z: https://github.com/eligrey/FileSaver.js.

20. *Moment/Moment* [online]. 2021-04-06 [cit. 2021-04-06]. Dostupné z: https://github.com/moment/moment.

21. *Změna URL a skrývání fbclid pomocí JavaScriptu* [online] [cit. 2021-04-06]. Dostupné z: https://www.michalspacek.cz/zmena-url-a-skryvani-fbclid-pomoci-javascriptu.

22. ARRACHEQUESNE, Damien. *Introduction* [online]. 2021-03-21 [cit. 2021-04-06]. Dostupné z: https://socket.io/docs/v4/index.html.

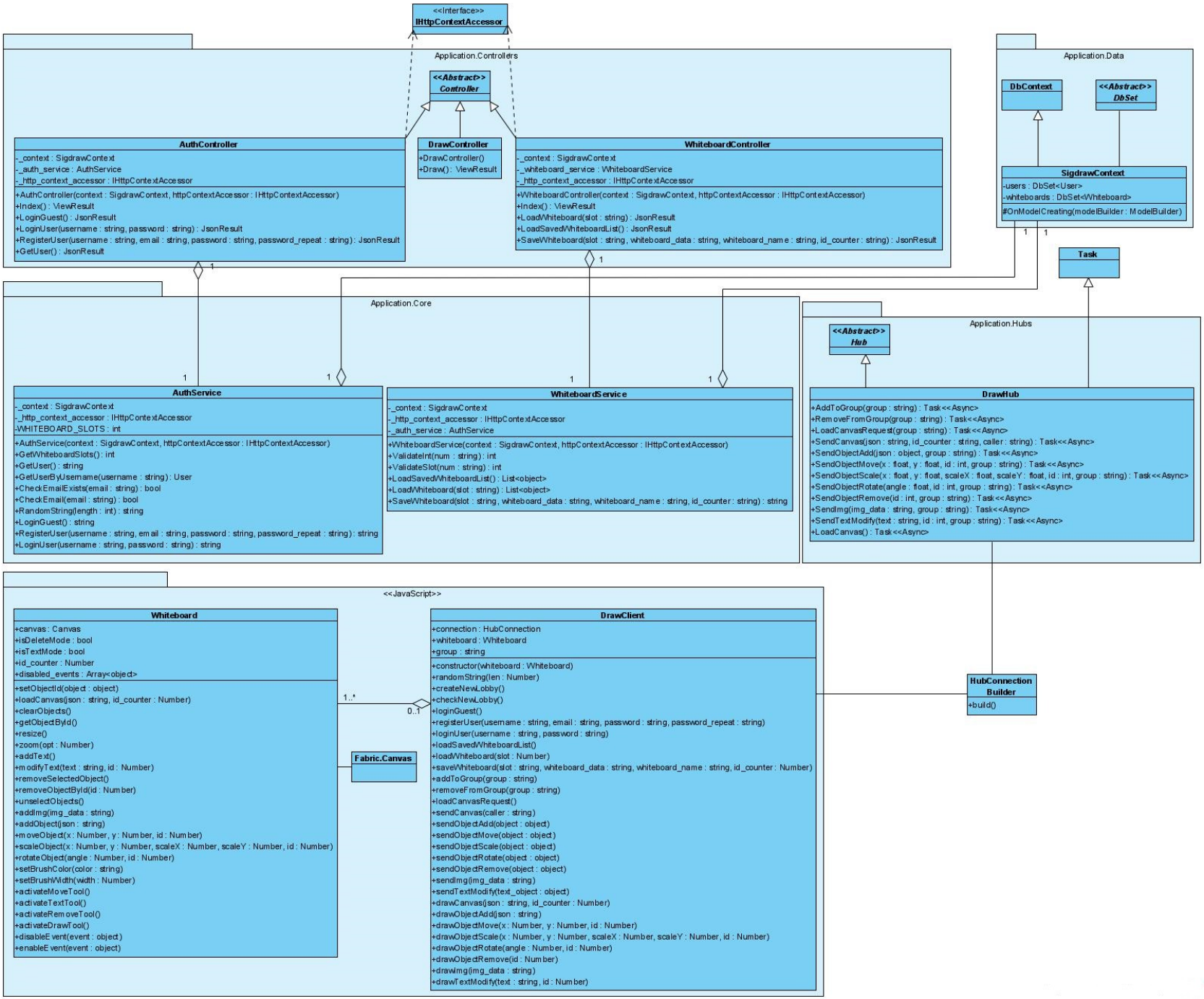
23. *oCanvas - Object-Based Canvas Drawing* [online] [cit. 2021-04-06]. Dostupné z: http://ocanvas.org/.

24. LAVRENOV, Anton. *Konva.Js - JavaScript 2d Canvas Library* [online]. 2021-03-02 [cit. 2021-04-06]. Dostupné z: https://konvajs.org/index.html.

25. ARDALIS. *Přehled ASP.NET Core MVC* [online] [cit. 2021-04-06]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/mvc/overview.

Příloha A

Třídní diagram aplikace



Obrázek A.1: Třídní diagram aplikace

Příloha B

Návod ke spuštění aplikace

Aplikace je ve formě projektu pro vývojové prostředí Visual Studio (vyvíjena ve verzi Commnunity 2019 v16.8.2).

1. Aktualizujte vývojové prostředí Visual Studio na nejnovější verzi.
2. Zkontrolujte, zda je instalace vývojového prostředí Visual Studio rozšířena o sadu funkcí "Vývoj pro ASP.NET a web". Pokud tomu tak není, doinstalujte tuto sadu funkcí.
3. Otevřete projekt standardním postupem pro Visual Studio.
4. Spusťte projekt použitím profilu IIS Express.

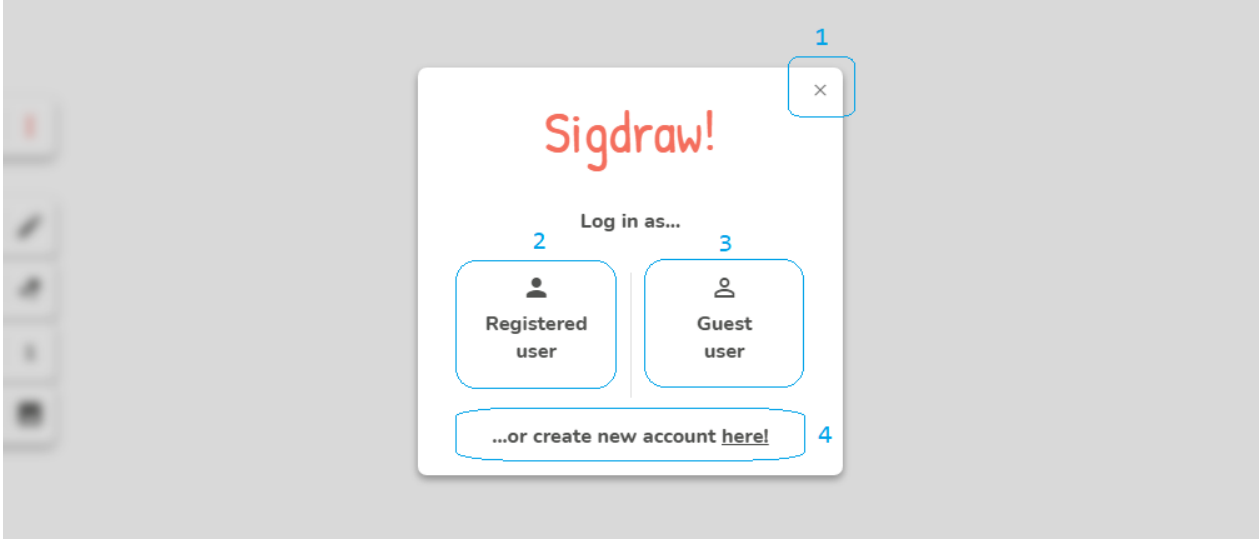
Je také možné přímo navštívit webovou stránku aplikace, která je v době odevzdání práce zveřejněna na testovací URL adrese `http://laj0015test.aspifyhost.cz`.

Příloha C

Manuál k aplikaci

V tomto manuálu jsou popsány veškeré ovládací prvky aplikace Sigdraw. Ke každé sekci připadá obrázek, ve kterém jsou čísly vyznačeny určité ovládací prvky, a seznam s vysvětlením jejich funkcí. Ovládací prvky, které se v aplikaci vyskytují častěji, jsou popsány pouze jednou.

C.1 Okno volby přihlášení



Obrázek C.1: Okno volby přihlášení

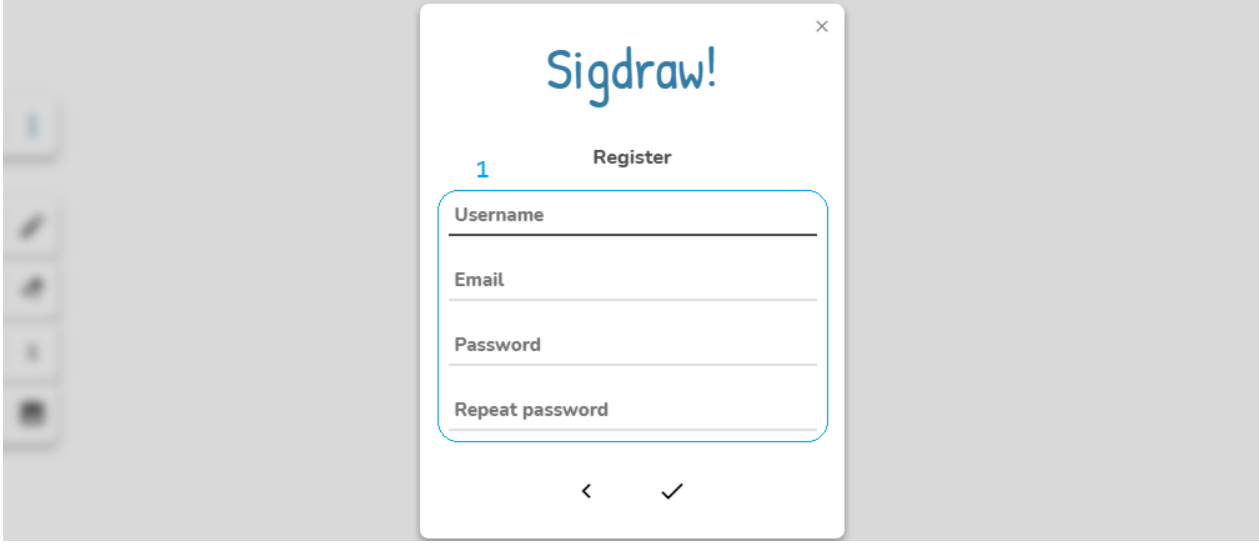
1. **Zavírací tlačítko** - slouží k uzavření vyskakovacího okna.
2. **Tlačítko přihlášení registrovaného uživatele** - zobrazí přihlašovací formulář.
3. **Tlačítko přihlášení neregistrovaného uživatele (hosta)** - přihlásí uživatele jako hosta a zavře okno volby přihlášení.
4. **Odkaz na registraci** - zobrazí formulář k registraci nového uživatele.

C.2 Okno přihlášení registrovaného uživatele

Obrázek C.2: Okno přihlášení registrovaného uživatele

1. **Přihlašovací formulář** - do textových polí tohoto formuláře uživatel vyplní své přihlašovací údaje. Při vyplnění nesprávných údajů bude uživatel aplikací informován pomocí varovné zprávy.
2. **Tlačítko zpět** - zobrazí předchozí sekci vyskakovacího okna.
3. **Potvrzovací tlačítko** - potvrdí danou akci.

C.3 Okno registrace



Obrázek C.3: Okno přihlášení registrovaného uživatele

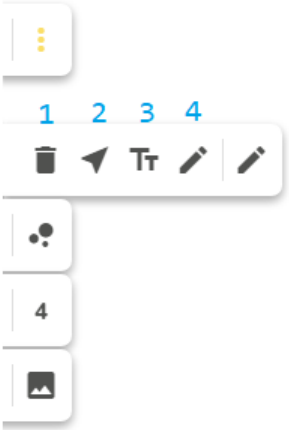
1. **Registrační formulář** - do textových polí tohoto formuláře uživatel vyplní své registrační údaje. Při vyplnění nesprávných údajů bude uživatel aplikací informován pomocí varovné zprávy.

C.4 Nabídka možností

Obrázek C.4: Nabídka možností

1. **Ikona profilu** - zobrazí okno volby přihlášení.
2. **Ikona složky tabulí** - zobrazí okno uložení / načtení tabulí (pouze pro registrované uživatele).
3. **Ikona plus** - zobrazí potvrzovací okno s výzvou, po jejímž potvrzení je vygenerován nový odkaz sdílení a uživatel je odpojen z aktuální sdílející skupiny.
4. **Ikona sdílení** - zkopíruje URL adresu (odkaz sdílení) do schránky.

C.5 Nabídka nástrojů

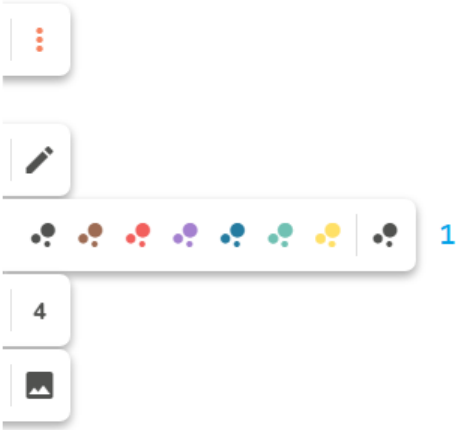


Obrázek C.5: Nabídka nástrojů

1. **Ikona koše** - aktivuje režim mazání objektů. Následným klikáním na objekty tabule dojde k jejich smazání.
2. **Ikona šipky** - aktivuje režim výběru. Objekty lze vybírat následným klikáním na ně. Vybrané objekty lze transformovat nebo přesouvat.
3. **Ikona textu** - aktivuje nástroj vkládání textu. Text lze vkládat kliknutím na požadovanou pozici na tabuli. Pro úpravu již vloženého textu je nutné na něj dvakrát kliknout v režimu výběru.
4. **Ikona tužky** - aktivuje režim volného kreslení. Čáry lze kreslit držením levého tlačítka myši a tažením po tabuli.

Nástroj přiblížení nebo oddálení plátna tabule není nijak graficky reprezentován. Plátno lze přiblížit nebo oddálit pomocí kolečka myši.

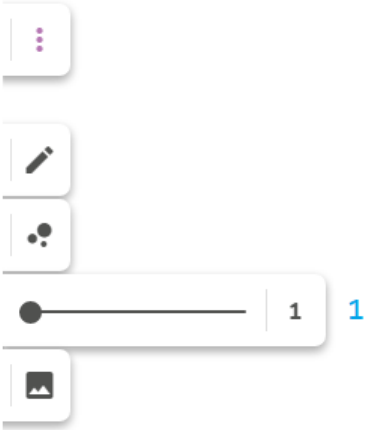
C.6 Nabídka barev



Obrázek C.6: Nabídka barev

1. **Ikona barvy** - nastaví danou barvu pro text a čáry. Aktuálně zvolená barva je zobrazená na pravé straně nabídky za dělící čarou.

C.7 Nabídka velikosti čáry / textu



Obrázek C.7: Nabídka velikosti čáry / textu

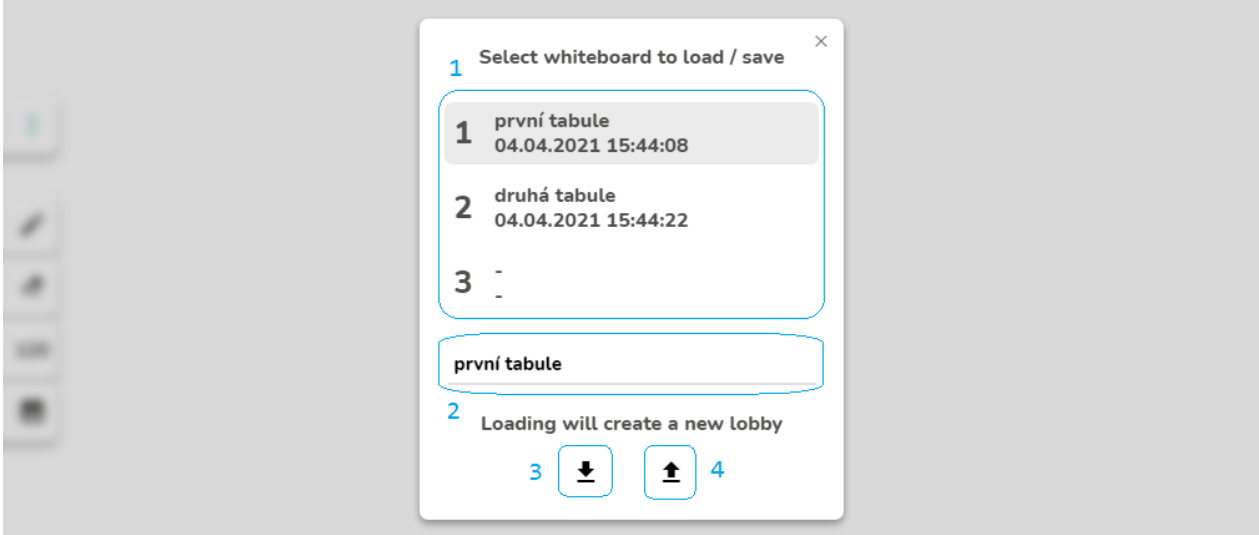
1. **Posuvník velikosti čáry / textu** - Dovoluje nastavit velikost čáry a textu na hodnotu mezi 1 až 120 pixely. Aktuálně zvolená velikost je zobrazena na pravé straně nabídky za dělící čárou.

C.8 Nabídka obrázku

Obrázek C.8: Nabídka obrázku

1. **Ikona šipky nahoru** - slouží k nahrání obrázku z místního úložiště na tabuli.
2. **Ikona šipky dolů** - slouží k převedení aktuálního pohledu na obrázkový soubor ke stažení.

C.9 Okno uložení / načtení tabule



Obrázek C.9: Okno uložení / načtení tabule

1. **Úložné pozice tabulí** - po kliknutí na danou pozici je pozice vybrána pro uložení nebo načtení.
2. **Formulář názvu tabule** - tabule bude uložena pod vyplněným názvem.
3. **Ikona šipky dolů** - nahraje tabuli uloženou v aktuálně zvolené úložné pozici a vytvoří novou sdílející skupinu a odkaz sdílení.
4. **Ikona šipky nahoru** - uloží tabuli na aktuálně zvolenou úložnou pozici v databázi.

C.10 Způsob používání aplikace

Při prvním navštívení aplikace má uživatel na výběr ze dvou způsobů přihlášení - přihlášení jako registrovaný, nebo jako host. Registrovaní uživatelé mají k dispozici funkci ukládaní tabulí do databáze na celkem tři úložné pozice. Hosté nemají přístup k této funkci, nemusí se však zatěžovat registrací. Pokud se uživatel chce zaregistrovat, může použít odkaz v okně volby přihlášení.

Po přihlášení má již uživatel k dispozici sdílenou tabuli, na kterou může společně s ostatními uživateli volně kreslit, vkládat text, obrázky a provádět další akce.

Uživatel sdílí tabuli s těmi uživateli, kteří se do aplikace dostali pomocí stejného odkazu sdílení. Ten jim může být zaslán jiným uživatelem. Odkaz sdílení je stejný jako URL adresa aplikace, s tím, že je k němu připojen řetězec identifikující aktuální tabuli. Odkaz sdílení lze získat kliknutím na příslušné tlačítko v nabídce možností nebo manuálním zkopírováním URL adresy.